La recherche d'informations sur le World Wide Web : utilisation des méta-informations dans une architecture de systèmes de recherche coopérants

Doan Bich-Liên

THÈSE

présentée par

M^elle Bich-Liên DOAN

pour obtenir le grade de Docteur
de l'Université Jean Monnet
et
de l'École Nationale Supérieure des Mines de Saint-Étienne
(Arrêté ministériel du 30 mars 1992)

Spécialité : Informatique

La recherche d'informations sur le World Wide Web : utilisation des méta-informations dans une architecture de systèmes de recherche coopérants

Thèse soutenue le 21 décembre 2000 devant le jury composé de :

Président :	Francis Rousseaux
Directeur de thèse :	Jean-Jacques Girardot
Rapporteurs :	Geneviève Lallich-Boidin
	Robert Mahl
Examinateurs :	Richard Baron
	Michel Beigbeder
	Marie-Jo Bellosta

Thèse préparée au sein du département **R**éseaux, **I**nformation et **M**ultimedia.
Ecole Nationale Supérieure des Mines de Saint-Étienne

THÈSE

présentée par

M^elle Bich-Liên DOAN

pour obtenir le grade de Docteur
de l'Université Jean Monnet
et
de l'École Nationale Supérieure des Mines de Saint-Étienne
(Arrêté ministériel du 30 mars 1992)

Spécialité : Informatique

La recherche d'informations sur le World Wide Web : utilisation des méta-informations dans une architecture de systèmes de recherche coopérants

Thèse soutenue le 21 décembre 2000 devant le jury composé de :

Président :	Francis Rousseaux
Directeur de thèse :	Jean-Jacques Girardot
Rapporteurs :	Geneviève Lallich-Boidin
	Robert Mahl
Examinateurs :	Richard Baron
	Michel Beigbeder
	Marie-Jo Bellosta

Thèse préparée au sein du département **R**éseaux, **I**nformation et **M**ultimedia.
Ecole Nationale Supérieure des Mines de Saint-Étienne

A mes parents,
à mes sœurs.

Remerciements

Je tiens ici à exprimer mes plus vifs remerciements à tous les membres du jury ainsi qu'à toutes les personnes qui, de près ou de loin, m'ont aidé durant ces années de thèse :

Monsieur Jean-Jacques Girardot, maître de recherche et directeur du département RIM de l'Ecole Nationale Supérieure des Mines de Saint-Etienne, pour m'avoir accueillie dans son laboratoire et pour la confiance qu'il m'a accordée tout au long de cette thèse.

Monsieur Francis Rousseaux, professeur à l'université de Reims, qui m'a fait l'honneur et le plaisir d'être président du jury de thèse.

Madame Geneviève Lallich, professeur à l'université de Lyon I et Monsieur Robert Mahl, professeur de l'Ecole Nationale Supérieure des Mines de Paris, qui m'ont fait l'honneur de rapporter ce travail.

Monsieur Michel Beigbeder, maître assistant à l'ENSMSE, pour m'avoir fait l'honneur d'être membre du jury, pour sa grande disponibiblité et pour ses conseils techniques.

Monsieur Richard Baron, maître de conférence à l'université de saint-Etienne et Madame Marie-Jo Bellosta, maître de conférence à l'université de Paris IX Dauphine pour avoir accepté de faire partie du jury.

Je tiens à remercier particulièrement Marie-Jo Bellosta, pour avoir si gentiment accepté de relire et de corriger une partie de ma thèse, mais aussi pour ses remarques constructives et pertinentes.

Je remercie tous les membres du département RIM, Annie, Mihaella, Roland, Philippe ainsi que tous les collègues de léquipe : Bernard, Bertrand, Fernando, Pierre, Faiza, Yamina, Camille et Gildas pour m'avoir témoigné leur sympathie et pour nos discussions philosophiques ;-)

Je remercie les secrétaires du centre, Liliane, Marie-Line et Zahia pour leur aide efficace et pour leur gentillesse.

Je remercie Hervé, Saliha, Stéphane, Gilbert, Jean-Pierre, Florence, Michel, Sophie, Valérie, Pascal, Laurent, Hubert, Franck et tous ceux que je n'ai pas nommés pour leur présence et leur soutien.

Je remercie tout particulièrement Mahdi, pour l'aide qu'il m'a apportée au cours de cette thèse et pour son soutien indéfectible.

Je tiens à exprimer toute ma reconnaissance à ma famille, pour son soutien inconditionnel de tous les instants et pour avoir toujours été présente dans les moments importants de ma thèse.

Deux types d'outils de recherche sont actuellement utilises pour aider l'utilisateur à trouver des informations sur le Web : les moteurs de recherche (Google) et les annuaires thématiques (Yahoo). Cependant les réponses sont fortement entachées de bruit pour les outils universels, et de silence pour les outils thématiques. De plus, le problème qui se pose est de suivre l'augmentation constante du volume de pages Web : la scalabilité. Pour réduire le bruit et le silence nous introduisons un niveau logique avec la notion de document Web au dessus du niveau physique matérialisé par les pages Web. Les documents Web sont organisés en DAG (Directed Acyclic Graph) et sont décrits par des méta-informations. Dans la hiérarchie de documents, nous utilisons la technique de propagation des attributs de méta-informations le long de la hiérarchie des documents. Ceci nous permet de diminuer à la fois le bruit et le silence en combinant des recherches qui portent sur les attributs de méta-informations avec la recherche traditionnelle dans le texte intégral, tout en exploitant la structure logique des documents Web. Pour le problème de scalabilité, nous proposons une architecture fondée sur deux nouvelles classes d'outils de recherche. Les outils généralistes ont vocation à parcourir, indexer et connaître tout le Web mais d'une façon superficielle, ils sont par exemple capables d'indiquer tous les sites dont un des domaines concerne l'environnement. Les outils spécialistes ont pour but de collationner et d'indexer toutes les pages de tous les sites d'un domaine de connaissance particulier (par exemple, l'environnement). Nous proposons enfin un modèle de dialogue entre ces nouveaux composants permettant de fournir un service global qui adresse à la fois les problèmes de bruit, de précision et de scalabilité. Un spécialiste devient à son tour un document qui s'auto-décrit et participe à l'architecture des systèmes coopérants.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Aujourd'hui, le World Wide Web (toile d'araignée mondiale), communément appelé Web, constitue un important gisement d'informations, dont l'originalité et le succès tiennent à la fois à la convivialité de l'interface d'accès, et à la facilité de publier ses propres documents.

Le succès du Web se manifeste jour après jour par la croissance exponentielle du nombre de serveurs et de pages HTML, par l'augmentation du nombre d'internautes [1], et par les nombreux services disponibles sur le Web tels que l'annuaire téléphonique, le commerce électronique, les enchères à distance, les conseils juridiques, le télé-enseignement, et l'analyse financière. Pour illustration en figure 1.1, d'après les sources NEC Research, IDC et [HA99] il y avait 300 millions de pages Web en 1997, 900 millions en 1999 et plus de de trois milliards de pages en 2000.

Compte tenu des critères économiques tels que la productivité, l'économie des marchés et la mondialisation des échanges, il est devenu indispensable de retrouver le plus rapidement possible des informations pertinentes. La recherche d'informations sur le Web est devenu un des domaines les plus abordés dans les conférences internationales (cf. VLDB2000, ICDT2001, DL2000, CIKM2000, IJCAI'99).

Dans la suite de ce chapitre nous décrivons la problématique de cette thèse, nos objectifs et notre proposition. Nous terminons ce chapitre par l'organisation de la thèse.

[1] nom donné aux utilisateurs du Web

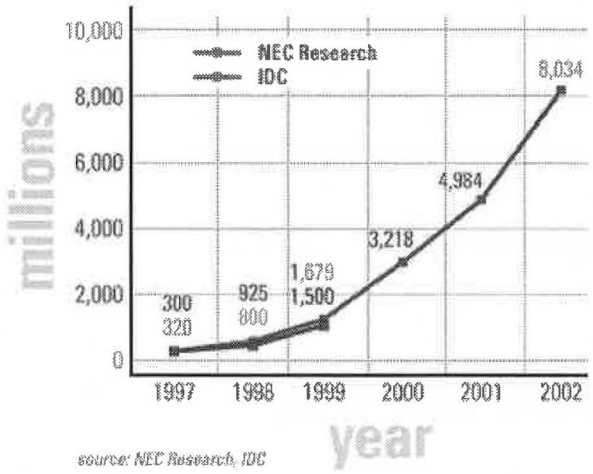


FIG. 1.1 – Evolution du Web en nombre de pages

1.1 Problématique

À ses débuts, le Web ne proposait qu'un nombre restreint de sites, majoritairement hébergés par des organismes universitaires, et le réseau avait un débit largement suffisant pour un petit nombre d'internautes. De ce fait, les sources et les quantités d'informations étaient peu importantes et en conséquence faciles à trouver puisque le nombre de pages était alors suffisamment restreint pour que l'on puisse parcourir les réponses rapidement.

Aujourd'hui les données du problème sont différentes en terme d'échelle, la quantité d'information qui circule sur les lignes du réseau augmente sans cesse. Nous sommes face à un problème de surcharge de l'ensemble des acteurs (producteurs, infrastructures, utilisateurs), qui est un frein à l'efficacité dans le processus de la recherche d'informations.

Pour répondre à cette demande croissante, les premiers systèmes de recherche d'informations (SRI) sont apparus sur le Web. Un SRI est un système qui fournit des réponses parmi un ensemble de documents face à un besoin d'informations émis par un utilisateur.

Nous détaillons les trois problèmes rencontrés par les SRI et que nous adressons dans cette thèse : la non-pertinence des réponses fournies par les SRI, l'insatisfaction des outils face à l'hétérogénéité et au manque de structuration du Web, ainsi que les difficultés dues à la centralisation de l'indexation et de la

recherche.

- **bruit et silence**

  La non-pertinence des réponses fournies par les SRI par rapport au besoin d'informations d'un utilisateur est dûe aux problèmes de synonymie, de multi-linguisme et d'ambiguité des termes. Plus précisément, la non pertinence est caractérisée par les notions de *bruit* et de *silence* [Rij79], illustrés en figure 1.2. Le bruit est défini par le nombre de documents retrouvés et non pertinents à la requête, tandis que le silence est défini par le nombre de documents non retrouvés par le système mais pertinents à la requête.

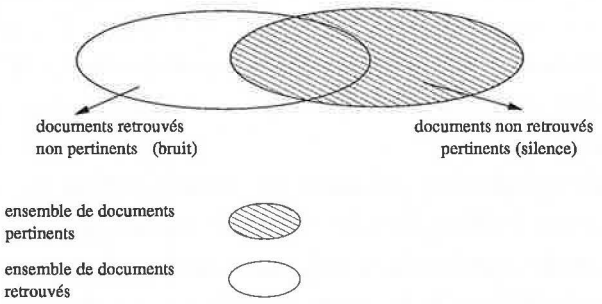


FIG. 1.2 – Bruit et silence

Actuellement ce problème n'est résolu par aucun outil de recherche. Considérons les moteurs de recherches et les annuaires thématiques. Le moteur de recherche effectue le plus souvent une recherche en texte intégral. Au préalable l'outil a constitué une base d'index en associant des mots-clés aux pages contenant ces mots-clés. L'utilisateur exprime sa requête sous forme de mots-clés, séparés ou non par des opérateurs booléens, et le moteur filtre les pages contenant une combinaison de ces mots-clés. L'annuaire thématique effectue une recherche exploratoire qui procède par un parcours d'un arbre de concepts fixes de plus en plus spécialisés jusqu'à trouver un thème précis.

Par exemple, considérons un utilisateur non spécialiste qui cherche tous les sites Web présentant un panorama des problèmes de pollution liés à l'environnement. Il pose la requête "environnement" à un moteur en texte intégral comme AltaVista. Parmi le million de réponses obtenues sous forme d'une liste de références, se trouvent des pages dans des domaines très divers. En effet le mot "environnement" peut se trouver dans des expressions comme "environnement informatique", "l'environnement in-

terstellaire", "l'environnement du travail", et n'est pas seulement lié au domaine de l'écologie. Dans ce cas, il y a beaucoup de bruit. Les réponses sont difficilement exploitables, ne serait-ce que parce que l'on imagine mal un utilisateur parcourir toutes les pages résultats. Il pose la même requête à un annuaire thématique sur l'environnement. Il trouvera certainement des réponses pertinentes, mais loin d'être exhaustives ! Ce type de recherche se traduit par peu de bruit mais beaucoup de silence. Considérons maintenant un utilisateur expert qui utilise une liste de mots-clés pertinents, par exemple "floculation, eau" pour faire une recherche sur Altavista. Il a à faire face à un très grand nombre de réponses avec beaucoup de bruit comme précédemment. Cependant, s'il précise le contexte, par exemple "environnement" dans sa requête avec l'opérateur "et", il risque d'y avoir du silence, simplement parce que ce contexte n'est pas pris en compte par le moteur.

- **hétérogénéité et manque de structuration** Le deuxième problème porte sur l'hétérogénéité et le manque de structuration explicite du Web. Les outils de recherche actuels sont aveugles à la structure et à la sémantique des sites Web qu'ils indexent. Il y a donc une perte d'information (contexte, structure) au moment de la collecte des pages Web qui se répercute sur l'indexation jusqu'au moment de la requête.

  La problématique devient la suivante : que faire pour qu'un moteur de recherche puisse comprendre la structure et la sémantique d'un site Web sans avoir à parcourir aveuglément toutes les pages d'un site ?

- **centralisation**

  Le troisième problème est celui de la centralisation de l'architecture des systèmes de recherche. Les outils de recherche centralisés sont limités face au problème de la "scalabilité", c'est-à-dire leur capacité à rester performants lorsque la quantité d'information augmente ou évolue. Dans le cadre du Web, ce problème se traduit par des délais d'exploration et des délais de rafraichissement d'index qui amènent une perte de qualité de l'information. Par exemple, Alta Vista et Google admettent un délai de rafraîchissement de l'index d'au moins 6 semaines, ce qui est énorme compte tenu de l'évolution du Web. Entre deux rafraichissements, certains mots utilisés dans la requête et contenus dans des pages n'apparaissent plus dans ces pages mais sont toujours référencés dans les index [And00]. L'image réelle du Web est différente de l'image du Web indexé rendue aux utilisateurs. L'évolution du Web se traduit aussi par des changements d'URLs

de pages, à l'origine de l'erreur 404, File not Found ? C'est un facteur de bruit et de silence.

De plus, l'explosion volumétrique de l'information sur le Web montre les limites des outils centralisés, dans la mesure où la taille de l'index ne peut plus couvrir la totalité des informations disponibles sur le Web, ce qui contribue à provoquer du silence. Par exemple, Google indexe 620 millions de pages en novembre 2000 et détient la plus grosse base d'index mais ne couvre que 20 pour cent du Web [Inc00], [ddwi00].

Dans cette thèse, nous proposons une architecture pour faire coopérer des outils de recherche existants en leur communicant des méta-informations afin de réduire le bruit et le silence, d'ajouter de l'information caractérisant les sites, et de distribuer l'indexation et la recherche afin de répondre au problème de la scalabilité du Web.

1.2 Objectifs de la thèse

Nos objectifs sont : l'indépendance physique, l'indépendance logique, la réduction du bruit et du silence, la scalabilité et l'utilisation des systèmes de recherche existants.

- **Indépendance physique** : l'accès aux informations doit être indépendant de la structure (par exemple qu'il s'agisse d'une base de données interrogeable via le Web, des pages HTML ou d'un cgi-script) et de l'organisation des données sur le Web. Une interface simple doit permettre à un utilisateur qu'il soit expert ou non d'exprimer son besoin d'information sous la forme d'une requête en simple texte ou structurée.
- **Indépendance logique** : à partir d'une structure logique existante, il doit être possible via des opérateurs de construire d'autres vues ou représentations logiques.
- **Réduction du bruit et du silence** : le système doit rendre des réponses pertinentes par rapport à la question et l'information doit être présentée sans redondance et exploitable. Elle doit être classée par ordre d'importance et lisible, c'est-à-dire que l'utilisateur doit avoir une idée du contenu de l'information donnée en réponse sans avoir à télécharger le document contenant cette information.
- **Scalabilité** : l'architecture doit être "scalable". Il faut fournir une architecture de systèmes de recherche d'informations capable de faire face à

l'explosion de l'information sur le Web, c'est-à-dire dont les performances en qualité des réponses et en rapidité (respectivement efficience et efficacité) restent identiques quel que soit le volume d'informations traitées.

- **Utilisation des systèmes existants** : notre système de recherche doit préserver l'existant, moteurs de recherche et annuaires sur le Web, et l'intégrer dans la nouvelle architecture. Cette architecture doit faciliter l'échange des informations entre les moteurs de recherche, et minimiser la consommation en bande passante en tirant profit des index existants.

1.3 Approche développée dans la thèse

L'originalité de l'approche développée dans la thèse repose sur l'introduction des documents structurés dans le Web et la proposition d'une architecture de systèmes coopérants prenant en compte les documents structurés et les outils de recherche existants.

- **les documents structurés** : Dans la recherche d'informations [Kor97], un large contenu textuel (par exemple un livre) est plus volontiers assimilé à l'ensemble de documents qui le composent. Par exemple un paragraphe, un chapitre ou encore une image d'un livre sont considérés comme des documents. La notion de document est étendue à toute information stockée sous n'importe quelle forme, par exemple les messages électroniques, les fichiers de données etc... Les documents formant le corpus homogène sont indexés. Or il s'est avéré que l'unité d'information prise en compte lors de l'indexation ne correspond pas forcément au document tel qu'il est extrait du corpus mais peut correspondre à des fragments ou à des composants du document qui seront alors indexés séparément : c'est pourquoi nous employons le terme document qui désigne à la fois l'élément traité et indexé dans un SRI mais aussi l'élément qui sera donné en réponse à une requête d'un utilisateur. Nombre de travaux prennent en compte ce découpage afin de fournir des réponses à un niveau de granularité adéquat, plus satisfaisantes pour l'utilisateur [HP93], [SA94]. L'importance de la structure dans les systèmes hypertextes est relevée dans différents projets que nous avons détaillés en annexe C. Le problème est que le découpage conduit à la création de nouveaux documents qui ne sont pas reliés entre eux. Il y a une perte de contexte associé au fragment d'information retenu conduisant à une perte d'information : supposons qu'un document soit découpé

en titre, résumé et paragraphes, si chacun de ces composants structurels est indexé séparément, lorsque l'interrogation porte uniquement sur un paragraphe, les informations présentes par exemple dans le titre du document n'ont pas été prises en compte par le SRI et répercutées dans l'index de ses composants. Cette insuffisance a été relevée et résolue dans les approches de [Khe95],[CK97], [Fou97], [Fou98], grâce à la propagation des informations par les liens de structure.

En utilisant les documents structurés dans le Web, les avantages apportés sont de différentes sortes :

- faciliter la production de contenu en fournissant un cadre structuré,
- faciliter le stockage et l'automatisation de l'indexation grâce au découpage logique de l'information,
- faciliter le répérage et la recherche de parties répondant à une requête.

Dans cette thèse, nous nous intéressons tout d'abord à fournir une nouvelle structuration des pages ou des collections de pages Web, qui puisse être exploitée par des outils de recherche. Pour cela nous introduisons le concept de *document Web* comme étant un ensemble de pages traitant d'un sujet précis.

Comme langage de description des documents Web, nous avons choisi XML qui est adapté à la manipulation des données structurées et semi-structurées. Nous définissons des méta-informations permettant d'interroger un site afin de déterminer s'il est pertinent à une requête. Avec XML, nous pouvons prendre en considération les liens entre documents et méta-informations en vue d'obtenir une meilleure qualité de réponses.

Le résultat de cette approche met en place un nouveau type de moteurs combinant les approches des annuaires thématiques avec celle des moteurs de recherche en texte intégral.

- **une architecture coopérante :**

Dans un deuxième temps, pour faire face à l'explosion de la quantité d'information sur le Web, nous proposons de distribuer les services de collecte, d'indexation et de consultation.

Grâce au modèle de documents Web, les sites sont explicités et les outils de recherche ne sont plus aveugles à leur sémantique. L'idée de base est que, ayant cette connaissance préalable des sites Web, certains outils de recherche spécialisés indexent une partie seulement des pages d'un site, par exemple les pages d'un domaine de connaissance particulier (l'environnement, la musique...). D'autres outils, à vocation générale n'indexent que

les méta-informations associées aux documents Web qui représentent en quelque sorte des résumés de pages Web. De plus, nous faisons en sorte que chaque outil ait une connaissance d'autres outils ou sites Web, c'est-à-dire qu'il stocke la référence à ces outils ou sites (l'URL en l'occurrence), ainsi qu'un descripteur associé à ceux-ci, extrait des méta-informations qui les décrivent. L'ensemble de ces outils, pour répondre à une requête d'un utilisateur, coopère par l'échange des méta-informations, afin de fournir une réponse plus précise, plus adaptée et dans son contexte. Ces outils de recherche coopérants permettent de faire face au problème de la "scalabilité" de la RI sur le Web, et d'apporter des réponses personnalisées et plus adaptées pour les différents acteurs s'engageant dans cette initiative.

La thèse se situe au carrefour des domaines de recherche suivants : les documents structurés, la recherche d'informations, les bases de données, les réseaux, les bibliothèques numériques, et la représentation des connaissances.

Les documents structurés [XML00], [Mil98] sont à la base de notre modèle de documents Web, en apportant la structuration et le format des documents. La recherche d'informations [Rij79], [Kor97], [BR99] nous a apporté les mécanismes d'indexation pour traiter des corpus textuels, ainsi que les modèles de correspondance permettant d'effectuer une recherche pour une requête donnée. Les réseaux et l'Internet [RFC93], [AL98] nous ont apporté les mécanismes de routage que nous définissons dans notre architecture coopérante. Des bases de données [OV91], [CBC+00] nous tirons les niveaux de représentation physique, logique et externe dans notre modèle de document Web, et le langage de requêtes sur des objets complexes. Les bibliothèques numériques [Les97], ont apporté les méta-informations [WGMD95] pour décrire les documents Web, ainsi que les langages classificatoires [Dew76], [Bet93a], [CDU98] pour classifier les sites Web. Le domaine de la représentation des connaissances a apporté les ontologies [Gua94] qui vont nous servir à définir la description d'un document.

1.4 Organisation de la thèse

Cette thèse s'organise comme suit :

Le chapitre 1 est l'introduction de cette thèse et présente la problématique générale de notre travail, à savoir comment retrouver des informations pertinentes sur le Web. Nous annonçons les grandes lignes de notre solution en

introduisant le concept de *documents Web* ainsi qu'une nouvelle architecture de systèmes coopérants.

Le chapitre 2 présente les concepts du World Wide Web indispensables pour appréhender la problématique de la recherche d'informations sur le Web : l'Internet, l'architecture client/serveur du Web et les caractéristique d'un serveur Web. Il présente aussi les technniques de routage et du DNS, sous-jacentes à notre architecture d'outils de recherche coopérants.

Le chapitre 3 est un état de l'art des systèmes de recherche d'informations sur le Web, vus selon deux axes : la représentation des données et l'architecture fonctionnelle. Les caractéristiques qui composent l'architecture fonctionnelle sont : le corpus, le gestionnaire d'index, l'interface utilisateur et le gestionnaire de requêtes. Cette étude nous permet de dégager les limites et d'argumenter en faveur de notre approche.

Au chapitre 4, nous proposons un modèle de structuration de documents qui permet à la fois de décrire les systèmes, les sites et les documents dans un format uniforme, en s'appuyant sur les notions de méta-informations. Nous montrons qu'introduire la structuration à travers les documents Web, ainsi que les méta-informations peut aider les moteurs de recherche dans leur phase d'indexation et de recherche. XML est un méta-langage adapté à la description des documents Web.

Dans le chapitre 5, nous spécifions une architecture qui introduit une coopération entre les systèmes de recherche d'informations grâce à l'échange des méta-informations. L'hypothèse sous-jacente à la définition de l'architecture est que la coopération entre les outils de recherche doit préserver les outils existants. L'architecture se compose de deux types de systèmes : les *s*pécialistes et les *g*énéralistes.

Dans le chapitre 6, nous concluons sur les apports de notre conception générale d'un système. En s'appuyant sur l'existant et sur une meilleure représentativité des sites Web, notre approche permettrait d'améliorer globalement les réponses des outils tout en diminuant le trafic sur l'Internet. Nous posons enfin les limites, qui se situent dans l'évaluation en grandeur réelle de notre approche et nous donnons les ouvertures et développements futurs de cette thèse. Nous suggérons notamment l'utilisation de notre architecture pour un système multi-agents dédié à la recherche d'informations sur le Web.

Chapitre 2

Présentation de l'Internet et du Web

Ce chapitre présente les caractéristiques du Web qui serviront de base pour l'élaboration de l'architecture fonctionnelle proposée. Le Web est défini comme un système hypermédia composé de structures associatives complexes, qui s'appuie sur le réseau Internet. Nous définissons tout d'abord les concepts liés à l'Internet : les adresses IP, le domain name server (DNS). Puis nous présentons les caractéristiques propres au Web : le protocole Hypertext Transfer Protocol (HTPP), l'architecture client/serveur, l'accès à l'information par navigation sur le Web, la représentation des informations sur le Web, la structuration et le Web, l'hétérogénéité et la dynamicité du Web.

2.1 L'Internet

L'Internet [RFC93] est un ensemble de réseaux interconnectés entre eux par des routeurs. Ces réseaux sont ceux appartenant à des entreprises, universités, administrations du monde entier. Ils sont reliés entre eux via des liaisons spécialisées à haut débit. Rappelons qu'un réseau informatique est un ensemble de machines connectées entre elles, généralement à l'aide de câbles ou de fibres optiques. Il n'y a cependant pas que des ordinateurs dans un réseau, mais aussi des dispositifs d'interconnexion comme des routeurs. Ces appareils ont pour but de permettre à deux réseaux ou plus de dialoguer entre eux.

2.1.1 Adresses IP (Internet Protocol)

C'est entre autres, grâce au protocole IP [RFC81] que fonctionne l'Internet. Ce protocole permet l'acheminement de paquets sur le réseau. Cependant, afin de permettre à une machine émettrice d'envoyer des paquets à une machine réceptrice, il est nécessaire de pouvoir les identifier de façon unique. Cette identification est réalisée grâce à une adresse IP. Chaque machine reliée à l'Internet doit disposer de sa propre adresse IP et celle-ci ne peut pas être utilisée par une autre machine.

Une adresse IP est structurée actuellement de la façon suivante : elle est composée de quatre nombres entiers, compris entre 0 et 255 (ces entiers sont codés sur 8 bits), séparés par des points.

Exemple : 192.34.56.78.

Les adresses IP ou Internet sont utilisées en particulier par le protocole IP afin de préciser, dans chaque paquet envoyé, l'identification de l'expéditeur et du destinataire. L'adresse du destinataire permet de choisir un chemin adéquat vers celui-ci, tandis que l'adresse de l'expéditeur peut être utilisée pour demander la retransmission de paquets perdus ou arrivés corrompus (c'est-à-dire que les informations qu'ils contiennent ont été modifiées de façon involontaire par le réseau, durant leur trajet).

La structure sur quatre entiers que nous venons de décrire ne concerne que la quatrième version du protocole IP (appelée plus couramment IPv4) ; c'est celle qui est actuellement utilisée sur Internet. Il faut cependant signaler que la nouvelle version d'IP, IPv6, est en cours de mise en place. Cette version apporte quelques améliorations techniques concernant le fonctionnement du protocole mais ce qui est le plus important, c'est la nouvelle structuration des adresses IP. Bientôt, on utilisera seize entiers au lieu de quatre pour spécifier l'adresse d'une machine.

Ces nouvelles adresses, compatibles avec les anciennes, ont été conçues afin de régler le problème de la pénurie d'adresses IP disponibles, résultante de la croissance exponentielle du réseau ces dernières années.

2.1.2 Le DNS (Domain Name System)

Des noms sont utilisés pour désigner les machines connectées à Internet, pour des raisons de simplicité d'utilisation. Ces noms sont appelés adresses DNS où DNS signifie Domain Name System (système par noms de domaines). Le système DNS [RFC87] permet donc d'attribuer un nom à une machine, sachant

qu'à chaque nom est associée l'adresse IP de la machine correspondante et que le système est hiérarchique (il peut être représenté par un arbre). Plus précisément, une adresse DNS est formée de plusieurs mots, comportant lettres ou chiffres, séparés par des points et ne comporte jamais d'espace ou de lettre accentuée. Le mot le plus général se situe à l'extrême droite de l'adresse DNS (on parle de Toplevel Domain Names), tandis que le plus précis (i.e. désignant une machine déterminée) est à l'extrême gauche.

Il est important de connaître l'existence des adresses IP que nous venons de présenter, mais en pratique, il est bien rare que nous ayons à les utiliser pour aller consulter un serveur Web. Notons que s'il est obligatoire d'attribuer une adresse IP à une machine connectée à Internet, il n'en est pas de même pour l'adresse DNS. Celle-ci est en effet facultative.

Prenons un exemple afin d'illustrer cette notion d'adresse DNS : `www.ibm.com`. Cette adresse DNS est celle d'une machine, appartenant à la société IBM, qui héberge le serveur Web de cette dernière. Comme nous venons de le voir, la partie la plus précise (c'est-à-dire désignant une machine) se trouve à gauche (www ici) tandis que la moins précise se situe à l'autre extrémité (com). Entre les deux, on distingue le nom de la compagnie IBM à laquelle appartient la machine considérée. On dit donc que cette machine, dont le nom est "www", fait partie du domaine "ibm.com".

L'extrémité droite d'une adresse DNS désigne le domaine le plus vaste, comme nous l'avons dit. On distingue, aujourd'hui, sept principaux domaines de plus haut niveau :

- .com, pour les entreprises (principalement américaines mais ce n'est pas une obligation),
- .edu, pour les organismes d'enseignement américains,
- .gov, pour les organisations gouvernementales des Etats-Unis,
- .mil, pour l'armée américaine,
- .org, pour les autres organisations,
- .net, pour les ressources propres du réseau,
- .int, pour les organismes internationaux.

En plus de ces domaines, qui concernent principalement l'Amérique du Nord, il en existe d'autres relatifs aux autres pays. Par exemple :

- .fr, pour la France,
- .uk, pour l'Angleterre,
- .de, pour l'Allemagne,
- .au pour l'Australie, etc.

Afin d'obtenir des informations sur les machines (correspondance entre adresses IP et noms de domaines), il existait un fichier appelé *hosts* qui centralisait toutes les descriptions des machines sur le réseau. Avec l'explosion du nombre de machines connectées, il a fallu trouver une autre solution. L'administration des machines devait être décentralisée. Le DNS est vu comme une base de données distribuée, permettant un contrôle local sur des portions de la base de données, chaque portion étant accessible sur le réseau global selon un schéma client-serveur. L'organisation hiérarchique des serveurs de domaines est similaire à l'arbre hiérarchique inversé des fichiers UNIX, et la racine de l'arbre est représentée par un point ".". Un domaine est un sous-arbre de l'espace des noms de domaines. Il peut être vu comme un ensemble de machines. Le DNS résoud le problème de nommage, chaque organisation a un nom unique donc peut créer les noms qu'elle veut. La tradition veut que le nommage des niveaux les plus hauts suive certaines règles et que les noms aient une certaine sémantique. Les données associées aux noms de domaines sont contenues dans des enregistrements de ressources. Chaque enregistrement appartient à une classe, liée à un type de réseau ou logiciel, par exemple l'Internet. Nous nous intéressons aux mécanismes de délégation, et de résolution de noms de domaines [?].

La délégation

Comment décentraliser l'administration ? En utilisant le mécanisme de délégation. Une organisation administrant un domaine peut le diviser en sous-domaines, dont chacun peut être administré par une autre organisation. Chaque organisation est responsable de la maintenance de toutes ses données (voir Fig. 2.1). Dans cette exemple, le domaine fr n'a qu'un pointeur sur les ressources du sous-domaine emse.fr, et emse.fr est administré par l'organisation emse qui gère les données comme elle le veut.

Les programmes qui stockent de l'information sur l'espace de nom de domaine sont appelés serveurs de nom. Ces serveurs de nom ont généralement une connaissance complète d'une partie de l'espace de nom de domaine, appelé zone, qu'ils chargent à partir d'un fichier ou d'un serveur de nom (dans ce dernier cas on parle de transfert de zone). On dit que le serveur de nom a une autorité sur une zone, voire sur plusieurs zones. Une zone contient les noms de domaine que le domaine de même nom de domaine contient, sauf les noms de domaines qui font parties des sous-domaines délégués. Dans la figure Fig. 2.2 la zone edu ne contient pas le nom de domaine mit. Les fichiers de zones sont des données sous

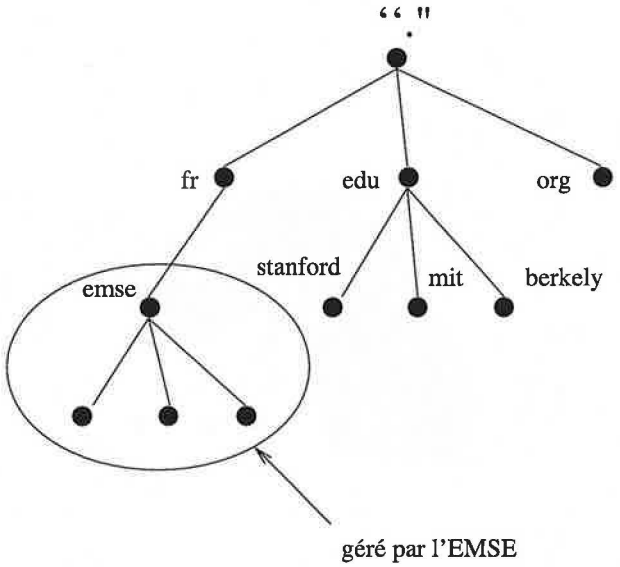


FIG. 2.1 – emse.fr est délégué à l'Ecole des Mines de Saint-Etienne.

forme de RR (resource records) qui obéissent à un certain RFC.

Le résolveur

C'est un client qui a accès aux serveurs de nom. Le résolveur peut questionner un serveur de nom, interpréter des réponses (enregistrements ou erreur), retourner des informations aux programmes qui ont émis la requête. Dans BIND, le résolveur est un ensemble de routines qui sont liés aux programmes tels que telnet, ftp. Il existe deux modes, le mode récursif et le mode itératif. Dans le mode récursif, le résolveur envoie une requête récursive à un serveur de nom. Ce dernier est obligé de répondre et s'il n'a pas l'information demandée, doit transmettre la requête à d'autres serveurs de noms. La requête peut alors être récursive ou itérative. Si elle est itérative, la réponse envoyée par les serveurs de noms est la référence à des serveurs plus proches du domaine demandé. Pour des raisons de performances, c'est le mode itératif qui est utilisé dans l'Internet.

Les caches sont également importants, un serveur peut être de type cache, il conserve pendant un temps défini par le TTL (*Time To Live*) les données sur les serveurs qu'il indexe. Un serveur peut être de type "forward" auquel cas il se comporte lui-même comme un client auprès d'autres serveurs de nom afin de propager la requête.

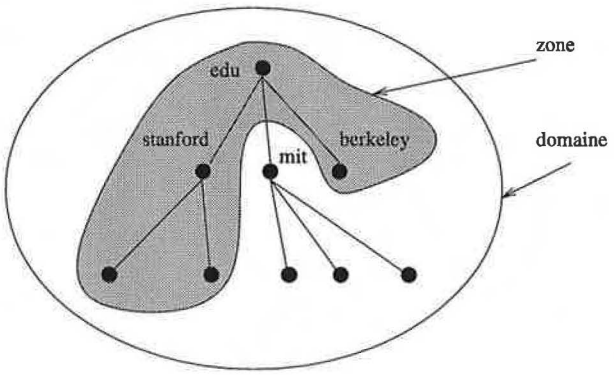


FIG. 2.2 – zone et domaine.

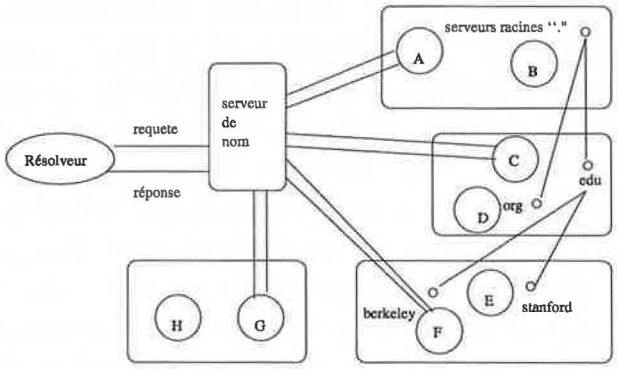


FIG. 2.3 – Processus du résolveur

Le DNS est une brique de l'Internet, c'est aussi le point de départ des protocoles de communication entre les systèmes de recherche d'informations définis dans l'architecture que nous proposons dans cette thèse.

Nous venons de définir les concepts de niveau physique, nous allons maintenant passer aux concepts du niveau application qui utilisent l'infrastructure physique de l'Internet pour communiquer.

2.2 Le Web

Nous déclinons les points suivants : l'architecture client-serveur, le système hypermédia, les méta-informations et l'environnement hétérogène et dynamique du Web.

2.2.1 Architecture du Web

L'architecture du Web est de type client/serveur. Pour clarifier ce concept, prenons une comparaison imagée : considérons un restaurant. Lorsqu'un client entre, celui-ci est "pris en charge" par un membre du personnel du restaurant, à savoir un serveur. Le serveur a pour rôle, comme son nom l'indique, de servir le client : il lui fournit un service. De plus, un serveur s'occupe généralement de plusieurs clients.

Sur un réseau comme Internet, il existe de nombreux serveurs, tels les serveurs Web, les serveurs de News, les serveurs FTP. Les serveurs Web fournissent des services à des clients, comme dans notre exemple du restaurant. Ces services correspondent à la mise à disposition de documents (textes, images, sons, etc.). Dans le cas des serveurs Web, les utilisateurs les consultent grâce à un programme, appelé client Web, navigateur ou butineur. De même, un serveur est capable de traiter plusieurs clients simultanément, comme dans notre exemple.

Le Web est un ensemble de serveurs proposant des documents accessibles via un protocole appelé HTTP pour HyperText Transfer Protocol [RFC96]. Un serveur Web est tout simplement un logiciel qui reconnaît ce protocole. Les programmes qui permettent de se connecter à ce type de serveurs, en utilisant donc HTTP, s'appellent des navigateurs Web (voir figure 2.4).

Serveurs Web

Un serveur Web met à disposition des utilisateurs de navigateurs des documents de tous types : textes classiques, images, sons, animations, applications

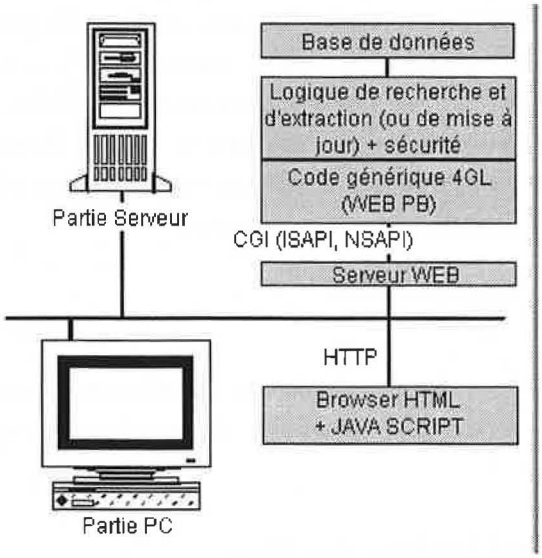


FIG. 2.4 – Architecture Web client/serveur.

Java ou même n'importe quel fichier binaire. Un navigateur est capable d'afficher un certain nombre de ces types de fichiers, en particulier tout ce qui est texte, image et son. Cependant, les documents qui structurent les serveurs Web sont un peu particuliers : ils sont appelés pages HTML. Le HTML (HyperText Markup Language) est un langage qui permet de composer un document, visualisé via un navigateur Web, et ayant une certaine mise en page.

Le HTML autorise l'incorporation dans une page de ce qu'on appelle des liens hypertextes (ou links en anglais). Ce sont ces liens qui permettent d'accéder aux différents documents (qui peuvent être d'autres pages HTML mais également tout type de fichier) proposés par un serveur web. Là où la technologie montre tout son intérêt, c'est qu'un lien peut désigner en réalité n'importe quel document mis à disposition par n'importe quel serveur Web accessible de par le monde. On dit alors qu'il s'agit de liens externes. Ces derniers créent une sorte de toile où les points d'intersection sont des documents HTML. Voilà pourquoi on parle de Web (toile en anglais).

Les URL Nous venons de le dire : un lien, inséré dans une page HTML, peut désigner n'importe quel document (au sens large, c'est-à-dire tout type de fichier) sur n'importe quel serveur. Ce système ne peut fonctionner que si l'on peut trouver un procédé permettant de donner un nom unique à chaque

document accessible. Ce procédé a pour nom : URL pour Uniform Resource Locator. Une URL a la forme suivante :

protocole ://adresse_serveur :numéro_de_port/chemin/document

Exemple : `http://www.emse.fr/products/index.html`

- Le protocole est par défaut HTTP. C'est le protocole utilisé pour permettre à un navigateur de dialoguer avec un serveur Web, comme nous l'avons vu plus haut. Mais les navigateurs reconnaissent d'autres protocoles, tel le FTP. En effet, un navigateur web permet d'accéder à de nombreux types de serveurs, y compris les serveurs FTP, Gopher, etc. Exemple : `ftp://ftp.emse.fr/` donne accès au serveur FTP de l'Ecole des Mines de St-Etienne.
- L'adresse du serveur peut être indiquée sous forme d'adresse IP ou DNS. En général, on utilise l'adresse DNS, plus facile à mémoriser et à utiliser. Le numéro de port est rarement indiqué. Il permet par exemple d'accéder à plusieurs serveurs sur une même machine (donc ayant la même adresse). Le port par défaut est le 80. Exemple : `http://www.emse.fr:80/products/index.html` est totalement équivalent à l'URL donnée en exemple plus haut. `http://www.emse.fr:8080/` désignerait un autre serveur fonctionnant sur la même machine.
- Le chemin, suivi d'un nom de fichier (ou de répertoire), indique précisément le fichier auquel on souhaite accéder, sur le serveur considéré. Remarquez que le nom d'un document HTML porte l'extension .html (ou parfois .htm). Lorsque l'URL ne se termine pas par un nom de fichier portant cette extension, mais par une barre oblique (/) c'est qu'il s'agit d'un nom de répertoire. Dans ce cas, c'est une page par défaut qui est généralement désignée. A noter également que seul le protocole et le nom du serveur sont obligatoires. Si on ne spécifie pas de nom de fichier, on obtient la page d'accueil (ou homepage) du serveur Web considéré, ou le répertoire racine du serveur FTP accédé.

Navigateurs Web

Les logiciels qui permettent de se connecter à des serveurs Web s'appellent des navigateurs Web (ou browsers en anglais), comme nous l'avons déjà dit. Les deux navigateurs les plus connus sont ceux de Microsoft (Internet Explorer) et de Netscape (Navigator dans la suite Communicator) en figure 2.5.

En 1993, un logiciel nommé Mosaic était la référence en matière de naviga-

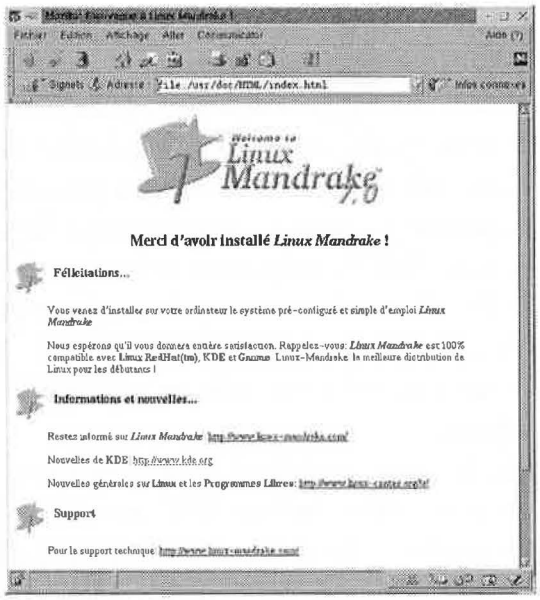


FIG. 2.5 – Navigateur Netscape

teurs Web. Il a été relégué au rang des antiquités depuis l'arrivée de Navigator puis d'Internet Explorer. Ces logiciels permettent non seulement d'accéder à des serveurs Web mais aussi à des serveurs FTP, Gopher, comme nous l'avons vu plus haut lorsque nous parlions des URLs.

2.2.2 Le Web, un système hypermédia

L'hypertexte est un outil pour construire et utiliser des structures associatives. Un document classique, par exemple un livre ou un article, a en général une structure linéaire, alors qu'un document hypertexte a une structure sémantique complexe, permettant de sauter d'un point à un autre, d'une idée à une autre, selon les centres d'intérêt de chacun. Cette structure permet de relier entre eux des mots, des fragments de texte, des documents. Les premiers systèmes hypertextes sont apparus avec Memex, système décrit par Vannevar Bush comme une sorte de bibliothèque privée mécanisée où un individu peut stocker livres, enregistrements et communications pouvant être consultées rapidement et de manière flexible. Xanadu (Ted Nelson 1965) reprend ce concept pour essayer de stocker toute la littérature dans le monde. Mais désormais, grâce à l'Internet, le Web a connu le plus grand succès pour sa convivialité, sa diffusion auprès d'un

plus large public par un accés aisé. Selon une remarque de Jacob Neilsen, dans une de ses études à Sun MicroSystems : "Plus les pages sont organisées, plus j'ai foi en leur apport d'information".

L'hypermédia généralise la notion d'hypertexte, dans ce sens où les éléments reliés entre eux sont du texte mais aussi des sons, des images, de la vidéo.

Le Web est un système hypermédia composé de ressources liées entre elles par des structures associatives. Les structures associatives permettent d'accéder à ces ressources par leur localisation unique, appelée Uniform Resource Locator (URL).

Les ressources sont des pages formatées en Hypertext Mark-up Language (HTML), des éléments multi-média, des scripts. Chaque page HTML peut contenir des fragments de texte, d'image qui pointent vers d'autres ressources du Web. Il est donc possible de passer d'une page à une autre au gré des liens définis dans les pages par leurs auteurs. Cette manière de parcourir les pages du Web se nomme le butinage.

Pour des raisons d'efficacité, il est recommandé de créer des pages réduites en nombre d'octets, de manière à accélerer leur téléchargement à partir de sites distants. Le découpage d'un document en pages HTML détermine sa structure physique. Cette structure est différente de la structure logique, qui elle est déterminée par l'imbrication ou le séquencement des parties du document.

HyperWave, une nouvelle génération du WWW

Dans HyperWave [Mau96], les documents sont structurés en hiérarchies de collections (à la façon de Gopher) avec des droits d'accès présentés en séquences de menus ayant une certaine portée (celle-ci est déterminée par les drois d'accès dans la gestion des utilisateurs). Un système UNIX fournit une hiérarchie de contextes (comme par exemple le thésaurus) si l'on prend soin des noms des différents nœuds de l'arbre. Chaque objet dans HyperWave appartient à un utilisateur et possède obligatoirement un titre, opionnellement un mot-clé. Pour créer des collections, il existe des clients, par exemple Amadeus ou Harmony. L'insertion des objets peut se faire manuellement dans des "CollectionHead". Un cluster regroupe des entités de même type (texte, ou son, ou vidéo etc...). L'utilisateur peut sélectionner des documents et les copier dans des collections, il s'agit de structurer l'information pour trouver l'information que l'on cherche rapidement, naviguer à travers l'espace d'information, fournir et maintenir ses propres données sur le WWW facilement (plusieurs utilisateurs peuvent stocker

leurs données sur un seul serveur), ceci permet à des utilisateurs travaillant sur les mêmes centres d'intérêt de partager plus facilement leurs informations puisqu'elles se trouvent au même endroit.

Le système admet des hiérarchies mutli-collections, c'est-à-dire qu'un document peut appartenir à plusieurs collections.

Autres systèmes hypertextes structurés

Dans la recherche d'informations classique, les documents sont homogènes en taille et en structure. Cependant, la RI est maintenant confrontée aux documents en texte intégral et hétérogènes, ainsi qu'aux différentes vues de l'utilisateur par rapport à certaines parties de documents. D'où le concept de "recherche de passage" [Kor95] (p.18) permettant de découper l'information à un niveau de granularité adéquat.Le WWW étant un système hypertexte, il est légitime d'étudier la recherche d'information dans les systèmes hypertextes. A la fin des années 80, les systèmes hypertextes ont commencé à émerger et la recherche d'informations et l'hypertexte se sont développés. Dans la gestion de l'information hypertexte, le document est découpé en fragments stockés et gérés dans un réseau de nœuds. Ces fragments sont de types très différents, par exemple, des fragments de texte tels que le résumé ou les références bibliographiques, des données gérées par un SGBD, comme la date de publication d'un document, une liste de termes représentant le contenu de l'information des documents de la collection, la définition d'un terme.

Méthodologies pour construire un système hypertexte

Nous nous intéressons maintenant de plus près à la structuration d'un ensemble d'informations hypertextuelles puis plus particulièrement à la structure d'un site Web. Il existe différentes méthodologies pour concevoir des systèmes hypertextes, par exemple l'une est basée sur la structure globale du document hypertexte, privilégiant les relations entre les pages, une autre analyse les relations entre les composants du document pour créer ce document, une autre technique consiste à créer les relations en se basant sur le contenu du document. Si nous nous focalisons sur l'aspect structurel de l'information nous distinguons plusieurs organisations possibles de l'information, chacune liée à un cheminement de lecture différent.

Structuration séquentielle Un livre est structuré de manière séquentielle, c'est-à-dire que sa lecture s'effectue de manière linéaire, avec peu ou pas de retour

en arrière. L'enchainement logique des chapitres lui confère une structure facile à suivre, rigoureuse et peu complexe, mais avec une pauvreté dans les relations associatives entre les idées et les sections.
Structuration en arbre Beaucoup d'informations sont structurées hiérarchiquement, par exemple l'organisation hiérarchique d'une société, l'arbre des connaissances. Les annuaires thématique tels que Yahoo ! proposent une vue hiérarchique et un classement par thème des différents sujets qui sont traités. L'avantage de cette organisation est que tout en permettant une vue synthétique de l'information et une couverture suffisamment riche en terme d'expressivité (relations de composition ou d'héritage pour parcourir des informations de plus en plus détaillées), elle n'est pas trop complexe pour engendrer une surcharge cognitive.
Structuration en toile Une toile est un graphe orienté où les nœuds représentent les informations et les arcs orientés sont les liens hypertextes entre ces nœuds. Le Web a une structuration en toile. La vision du Web est très "plate", c'est-à-dire que l'organisation des pages n'est pas apparente, même si elle existe (un ensemble de pages traitant d'un thème donné peut constituer un document Web). Les pages écrites par un même auteur ont généralement un point d'entrée repérable par une page d'accueil.

L' usage très libre des tags HTML entraîne une structuration très faible des documents. Les outils de recherche indexent les pages sans pouvoir tenir compte ni de la structure ni de l'existence des documents.

Afin de concevoir des documents sur le Web, des langages normalisés ont été créés et leur simplicité est à l'origine de leur succès sur le Web. Nous comparons les deux langages les plus importants afin de faire ressortir leurs forces et leurs limites.

HTML (Hypertext Markup Language)

HTML [LC95], [SM98], est issu du langage de structuration de documents SGML (Standard Generalised Mark-up Language). HTML offre un ensemble de balises normalisées et prédéfinies.

Prenons par exemple un extrait d'une HTML dans laquelle nous voulons faire figurer les coordonnées d'une personne :

```
<P>Bich-Lien Doan</P>
<ADDRESS>
Ecole Nationale Superieure des Mines de Saint-Etienne<BR>
158, cours Fauriel<BR>
```

```
F-42000 SAINT-ETIENNE<BR>
</ADDRESS>
```

Les balises <P>, </P> et <BR> marquent respectivement le début d'un paragraphe, la fin d'un paragraphe, et une fin de ligne. Les balises <ADDRESS> et </ADDRESS> marquent respectivement le début et la fin d'une chaîne représentant une adresse.

Notons que les balises HTML dans leur grande majorité, ont un rôle de mise en page (<P>, <BR>,...), c'est-à-dire définir l'apparence visuelle du texte. HTML permet de hiérarchiser des parties de documents, d'insérer des éléments multi-média, et de faire de la mise en page. Malheureusement, il est généralement utilisé abusivement pour la présentation de pages plutôt que pour la structuration de celles-ci. HTML est assez permissif, ce qui rend plus difficile aux parseurs de détecter les éléments structurels d'un document. La balise <ADDRESS> de l'exemple ci-dessus est un cas assez rare dans le langage HTML de balise sémantique qui permet de structurer le contenu de la page. Notons que la chaîne "F-42000" est un code postal au format officiel recommandé par la communauté européenne, le "F" représente l'identificateur du pays. Cependant le manque de structuration de ce document rend difficile un traitement automatisé de ce type de document, pour retrouver après indexation tous les documents mentionnant une adresse en France à PARIS.

Actuellement, HTML reste néanmoins le format hypertexte standard reconnu par tous les navigateurs du Web. Le méta-langage XML semble un candidat potentiel pour remplacer HTML dans l'avenir, car outre les qualités de séparer la structure et la sémantique de l'affichage d'un document, il semble s'imposer comme un standard de fait dans de nombreuses applications, il est actuellement reconnu par les navigateurs Internet Explorer et Netscape.

XML (eXtended Markup Language)

XML est le résultat de la coopération d'un grand nombre d'entreprises et de chercheurs du World Wide Web Consortium (W3C). Leur objectif est de définir un formalisme permettant d'échanger facilement des documents complexes sur le Web, en dépassant les limites imposées par HTML. La norme internationale existante pour l'échange de documents structurés, SGML (*Standard Generalized Markup Language*, ISO 8879 :1986) s'est révélée trop complexe pour les besoins du Web.

En résumé, le langage XML est issu des normes SGML et HyTime, développées

à l'origine avec comme cible les grands documents techniques, militaires ou civils. Dans XML, il n'a été retenu que les caractéristiques essentielles de ces normes, indipensables à une bonne utilisation sur le Web.

Revenons à l'exemple de la section précédente sur la représentation des coordonnées d'une personne. En XML, nous aurions quelque chose comme :

```
<AUTEUR>
    <PRENOM>Bich-Lien</PRENOM>
    <NOM>Doan</NOM>
    <ADRESSE>
        <ORGANISATION>Ecole Nationale Superieure des Mines</ORGANISATION>
        <RUE-NUM>158, cours Fauriel </RUE-NUM>
        <VILLE>SAINT-ETIENNE</VILLE>
        <CP>42000</CP>
        <PAYS code=''iso-3166''>F</PAYS>
    </ADRESSE>
</AUTEUR>
```

Cet exemple illustre certaines propriétés importantes du langage XML : la représentation de l'adresse en XML ne comporte absolument aucune indication sur sa mise en page. Rien ne dit par exemple, qu'il faut revenir à la ligne après avoir affiché le nom suivi du prénom. Les polices de caractères à utiliser pour les différents éléments ne sont pas spécifiées. Le code du pays est indiqué au format "iso-3166" mais peut apparaître dans le document final sous la forme "FRANCE" par exemple.

En résumé, de nombreuses propriétés graphiques ou typographiques sont absentes dans le source XML. Toutes ces propriétés sont définies dans un autre document : la feuille de style.

Cette séparation de la description structurelle des documents et de la description de leur représentation visuelle offre d'énormes avantages. On peut ainsi imaginer différentes représentations visuelles (des vues) d'un même document. Mais surtout, elle facilite le traitement automatisé de document. Ainsi, alors qu'il s'avérait difficile d'extraire à partir du document HTML automatiquement les différentes parties de l'adresse, il est très facile de récupérer à partir du formalisme XML ces adresses, et de les stocker dans une base de données en conservant la structure de ces données.

Pour conclure cette section sur le langage XML, citons un autre aspect du langage XML, à savoir son extensibilité. En effet, les balises utilisables dans le

source d'un document XML sont personalisables et sont définies dans un document séparé appelé DTD (Document Type Definition). Ainsi les utilisateurs peuvent définir librement leur propre structure de document. De plus la DTD permet d'échanger avec d'autres utilisateurs le format de la structure de document. Nous pouvons ainsi noter qu'il existe déjà des structures types de DTD qui garantissent que la sémantique des composants structurels — les éléments — des documents partagés est homogène. Par exemple MathML est un ensemble de balises pour XML permettant de représenter des formules mathématiques, l'extension SMIL permet de définir des présentations multimédia.

2.2.3 Les méta-informations

Les méta-informations sont les "informations sur les informations", c'est-à-dire des données permettant de caractériser et sélectionner de l'information. L'utilisation des méta-informations répond à différents besoins, par exemple, afin de sélectionner ou de filtrer des données particulières à un domaine, pour identifier des ressources ou évaluer la qualité, la pertinence d'une information (par exemple [PIC]).

Avec le succès de l'Internet et la progression de l'information en réseau, le travail sur les méta-informations est devenu stratégique, et a donné lieu à la première réunion de Dublin Core en 1995 [WGMD95] ; cette réunion a montré la nécessité de rassembler des spécialistes de différents domaines (informaticiens, documentalistes, linguistes...) pour discuter de la normalisation en information, parce qu'elle correspond à une vision d'avenir des échanges d'informations dans un contexte global (dont les institutions et les administrations). Dans un contexte distribué, il est nécessaire de standardiser les échanges d'information en provenance de services ou de bases de données hétérogènes.

Au cours du workshop de Dublin Core Metadata [SW95], il a été défini un nombre minimal de 13 éléments (DCMD) pour décrire la méta-information associée à un document sur le WEB. Ces 13 éléments sont :

- description
- title
- creator
- publisher
- otherAgent : personne qui a contribué au travail
- date
- resourceType : genre de ressource par exemple nouvelle, poeme, dictionnaire

- form : forme physique du document comme document HTML ou PostScript
- identifier : pour identifier de manière unique une ressource
- relation : relation à d'autres ressources
- source : origine des ressources
- language : langage du contenu de l'information
- coverage : localisation spatiale et caractéristiques temporelles de la ressource

Ces éléments sont représentés sous forme d'attributs-valeurs et peuvent avoir des "qualificateurs". Par exemple les attributs "Author" peuvent être qualifiés :
Attribute :Author
Value :(scheme=Email) doan@emse.fr

Attribute :Relation
Value :(scheme=URL, type=parent) `http://groseille.emse.fr/ENVTEST/~brodhag/projelev/summary0.html`

IAFA template [SW95] est un standard de méta-information pour l'Internet et a été utilisé pour décrire différents types de fichiers, accessibles par les protocoles HTTP, gopher, ftp, etc. Un ensemble de template définit la variété d'informations indexées pour décrire le contenu et les services fournis par les archives FTP. Un *file_type* identifie de manière unique un ensemble d'éléments. un fichier d'index comprend plusieurs types de templates. Actuellement, 14 template types sont utilisés : SITEINFO, LARCHIVE, MIRROR, USER, ORGANIZATION, SERVICE, DOCUMENT, IMAGE, SOFTWARE, MAILARCHIVE, USENET, SOUND, VIDEO, FAQ.

Les attributs peuvent avoir des versions (par exemple language_v0) être agrégés pour décrire un enregistrement (par exemple nom, prénom, adresse, tel), être identifiés de manière unique (*suffixe_handle*) indépendemment du *file_type*.

L'ensemble des spécifications décrites ci-dessus prennent en compte dans une certaine mesure la sémantique associée à une page (description du contenu de la page à l'aide de descripteurs de pages). Cette sémantique est traduite par un ensemble de champs de méta-information (standard Dublin Core) ou une collection d'attributs (IAFA templates) pour typer un document. Cependant, il n'apparait pas de notion de regroupement de pages et de description de ce regroupement de pages.

2.2.4 Le Web, un environnement hétérogène et dynamique

Dans cette section, nous abordons deux caractéristiques essentielles du Web, d'une part l'hétérogénéité du Web, et d'autre part la dynamicité qui soulève le problème de la scalabilité.

L'hétérogénéité du Web

L'hétérogénéité du Web porte sur les formats de documents, les types des ressources, leur contenu, les langues, la structuration des documents, leur fonctionnalité, les vues sur les documents. Nous relevons ci-après les différentes facettes de cette hétérogénéité qui lorsqu'elle n'est pas prise en compte, altère la qualité des réponses (et constitue une cause de l'augmentation du bruit et du silence).

- *Hétérogénéité des formats*
  Tous les formats de documents circulent sur le net, que ce soit du texte, des fichiers binaires, des fichiers de scripts, des documents multimédia, des bases de données. Par exemple, on peut trouver sur le net des fichiers formatés en HTML, en *postscript* (ps), en *portable data format* (pdf), en *Rich Text Format* (RTF), compressés en *gz* ou archivés en *tar*. Peu d'outils sont capables de reconnaître et d'interpréter ces différents formats. En général, ils reconnaissent le format image en filtrant par exemple les .gif ou .jpeg, comme c'est le cas pour Alta Vista.
- *Hétérogénéité des types*
  Les ressources du Web sont hétérogènes par leur type. Le type d'un document est sa définition en intention. Les documents d'un même type ont les mêmes caractéristiques. Par exemple un moteur de recherche est un document qui rend un service et est caractérisé par une interface de recherche. Un particulier peut publier sa page personnelle, caractérisée par son identité, ses domaines d'intérêt, son activité professionnelle, ses liens favoris sur d'autres pages Web. Un organisme public ou privé publiera des annuaires, des informations pédagogiques, grand public ou commerciales, journalistiques ou politiques. Notons que contrairement aux formats qui sont eux normalisés, différents types existent sur le net mais ne sont pas soumis à des standards de type.
- *Hétérogénéité de la nature de l'information*
  Les informations sont de différentes natures. Elles appartiennent à tous les domaines de la connaissance, par exemple les valeurs boursières, la haute

couture, l'informatique, l'art ....

- *Hétérogénéité des langues*
  Sur le Web, n'importe qui sur notre planète peut éditer une page sur la toile, dans sa propre langue. Le Web constitue un environnement multilingue.
- *Hétérogénéité de structure*
  Une autre caractéristique de l'hétérogénéité du Web est l'organisation de l'information. Les informations peuvent être sous des formes structurées très diverses, sous des formes semi-structurées ou non-structurées, par exemple les bases de données relationnelles, les données XML, les textes libres.
- *Hétérogénéité des fonctionnalités*
  Le Web ne propose pas simplement de l'information, et ne peut donc pas simplement être assimilé à une vaste bibliothèque mondiale. Le Web offre également des services. Un utilisateur peut accéder à des services en-ligne, par exemple commander une pizza sur le Web, réserver un avion en ligne, acheter des livres. Il peut aussi accéder à des services de recherche d'informations, grâce aux outils de recherche sur le Web ou grâce à des outils de recherche dans des bases de données spécialisées.
- *Hétérogénéité des vues*
  Les auteurs des pages sont eux-mêmes hétérogènes et publient selon autant de point de vues qu'il y a d'individus différents. De ce fait, les informations s'adressent à des publics différents. Les vues que chaque individu peut avoir d'une même source d'information sont différentes.

La dynamicité du Web

La dynamicité du Web est caractérisée par la croissance du nombre de sites et de pages Web ainsi que par la fréquence des modifications et des suppressions de pages ou de sites Web. [HA99], [HPPL98] tente d'évaluer l'augmentation du nombre de pages par site en analysant les données de deux outils de recherche (Infoseek et Alexa), et établit des lois mathématiques sur la distribution des pages par sites pour prévoir l'évolution future. [CF99] évalue la dynamicité du Web et traite du problème des pages dynamiques invisibles aux outils d'indexation.

Les modifications d'un fichier sont visualisables par comparaison de la date de dernière modification et de la date de création de ce fichier. La fréquence

des mises-à-jour de fichiers peut être traquée par les robots qui conservent la dernière date de modification d'une version de fichier ; cependant il est possible qu'une mise-à-jour ne soit pas prise en compte si la fréquence de téléchargement d'un site par un robot est inférieure à la fréquence de mise-à-jour de ce fichier. D'autre part, une page qui se crée n'est pas forcément détectée si elle n'est pas reliée à un site indexé.

Toutes ces approximations de calculs de fréquence et de volume d'information expliquent que les changements sur le Web sont difficilement quantifiables.

2.3 Conclusion

Dans ce chapitre, nous avons défini les mécanismes sous-jacents au Web : l'Internet, le DNS et la résolution des adresses IP grâce aux mécanismes de délégation. Nous avons présenté l'architecture client/serveur du Web, le protocole HTTP, les langages HTML, XML ainsi que l'intérêt des méta-informations. Plus particulièrement les caractéristiques d'hétérogénéité et de dynamicité du Web vont être prises en compte dans l'architecture que nous allons proposer. Ces deux caractéristiques justifient i) l'existence de moteurs spécialisés, ii) l'intérêt de trouver un langage commun pour la représentation et la structuration de l'information sur le Web, iii) l'architecture coopérante.

Elles sont à la base de notre étude des systèmes de recherche d'informations que nous allons comparer maintenant, afin de mettre en évidence leur difficulté à répondre à ces problèmes.

Chapitre 3

Etude comparative des systèmes de recherche d'informations (SRI) sur le Web

Dans ce chapitre, nous comparons différents systèmes de recherche d'informations sur le Web. Cette comparaison s'effectue selon deux grands axes :

- la représentation des données
- le type d'architecture déterminé à partir des critères de distribution, d'autonomie et d'hétérogénéité.

Les systèmes ont été choisis au regard de leurs fonctionnalités particulières et de leurs apports dans l'architecture fonctionnelle : Alta Vista et Yahoo ! sont respectivement représentatifs des systèmes de recherche en texte intégral et des annuaires thématiques ; HyPursuit et Aliweb sont étudiés pour le traitement du corpus et la représentation des données ; Northern Light pour la présentation des résultats ; Harvest, HyPursuit et InfoSleuth pour leurs architectures distribuées.

Une synthèse des systèmes de recherche d'informations étudiés permet de déterminer quelles caratéristiques manquent à ces systèmes pour atteindre les objectifs précédemment définis : indépendance physique, indépendance logique, réduction du bruit et du silence, scalabilité, et utilisation des systèmes de recherche d'informations existants.

3.1 Représentation des données

La représentation des données définit comment les données manipulées par un système sont structurées. Les SRI se distinguent par leur capacité à traiter différemment les documents initialement créés par leurs auteurs.

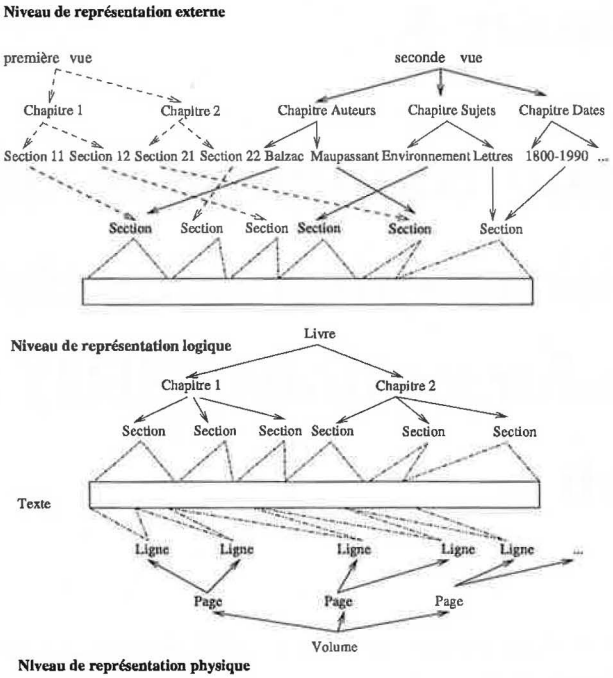


FIG. 3.1 – Niveau de représentation physique, logique et externe d'un livre

Nous distinguons trois différents niveaux de représentation : les niveaux de représentation physique, logique et externe :

- Le *niveau de représentation physique* d'un document est son découpage en unités physiques de stockage, c'est-à-dire sa décomposition en pages, en lignes, en fichiers.
- Le *niveau de représentation logique* appelé aussi structure logique est le découpage d'un document en différentes parties qui sont liées entre elles par des relations de séquence ou par des relations de composition.
- Le *niveau de représentation externe* est déduit d'une requête effectuée sur un niveau de représentation logique de base. Il permet de rendre une perception différente de l'utilisateur, appelée *vue*, par rapport à la structure de base, qui peut être par exemple celle d'un utilisateur par rapport à celle

d'un auteur.

Pour illustrer les différents niveaux de représentations, prenons l'exemple d'un livre en figure fig 3.1. Le niveau de représentation physique d'un livre décrit l'organisation du document sous forme de pages et de lignes, alors que le niveau de représentation logique est son découpage en chapitres, sous-chapitres, sections, sous-sections. Ce même livre peut être réorganisé selon d'autres critères, ici les sections sont regroupées d'après des critères de date, de sujet et d'auteur, il s'agit d'une représentation externe de ce livre.

Les trois niveaux de représentation des documents (physique, logique, externe) amènent à définir une caractéristique d'un SRI en fonction du niveau d'indépendance qu'il offre : *indépendance physique*, c'est la séparation entre les niveaux de représentation physique et logique d'un document, *indépendance logique*, c'est la séparation entre les niveaux de représentation logique et externe d'un document.

3.2 Architectures fonctionnelles

L'architecture fonctionnelle d'un SRI (fig. 3.2) est définie par l'identification de l'ensemble de ses composants, par la spécification de la fonction de chacun des composants, et par les interactions, et les contrôles de flux et de données entre ces composants. Elle comprend *l'interface utilisateur*, *le gestionnaire d'index*, *le gestionnaire de requête*, *le stockage* et *les méta-informations*.

Nous décrivons ci-dessous l'architecture fonctionnelle d'un SRI qui centralise les données et les traitements. Le gestionnaire d'index (1) s'occupe de la création des représentants des documents à partir du corpus et de méta-informations, ainsi que de la construction de la base d'index à partir de ces représentants. Les méta-informations sont utilisées à la fois par le gestionnaire d'index et par le gestionnaire de requêtes. Lors d'un besoin d'informations émis via l'interface utilisateur, formulé au moyen d'une requête et éventuellement de méta-informations (2), le gestionnaire de requêtes (3) traduit ce besoin en un langage compréhensible par le système. Puis le moteur de recherche interroge la base d'index avec les méta-informations, pour trouver les documents pertinents à cette requête grâce à une fonction de correspondance entre la requête et la base d'index. Les résultats sont alors présentés par l'intermédiaire de l'interface utilisateur.

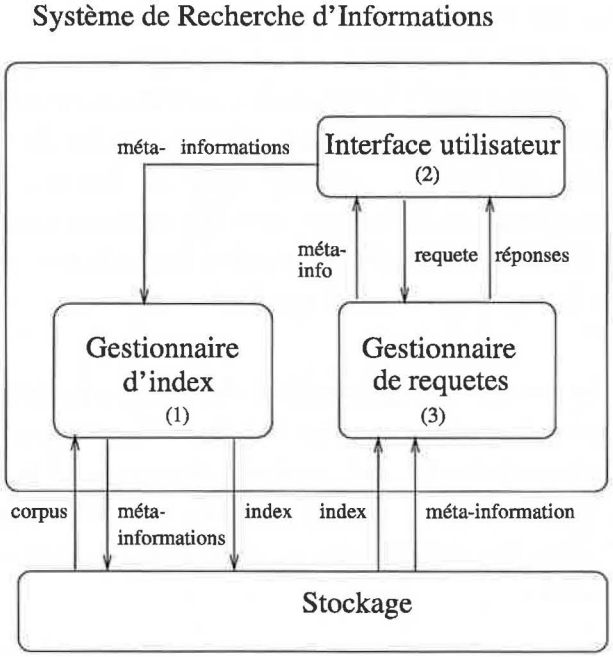


FIG. 3.2 – Architecture fonctionnelle d'un système de recherche d'informations

Nous étudions à présent plus précisément les composants de l'architecture fonctionnelle.

3.2.1 Composants d'une architecture fonctionnelle

Parmi les flux de données de l'architecture fonctionnelle, le corpus constitue le point d'entrée de tout système de recherche d'informations.

3.2.2 Le corpus

Le corpus représente l'ensemble des documents potentiellement interrogés et retrouvés par le SRI. Nous définissons quatre critères pour comparer les différents corpus des SRIs : l'hétérogénéité, le volume, l'évolution et le type de corpus.

a) L'hétérogénéité

L'hétérogénéité d'un corpus définit sa capacité à inclure des documents de domaine, de support ou de structure différents. Nous définissons l'hétérogénéité

d'un corpus comme une combinaison entre l'hétérogénéité structurelle, l'hétérogénéité physique et l'hétérogénéité sémantique où :

- L'hétérogénéité structurelle qualifie un corpus dont les documents n'ont pas la même structure logique.
- L'hétérogénéité physique qualifie signifie un corpus dont les documents sont de différents types (par exemple de type texte, image, multimédia).
- L'hétérogénéité sémantique qualifie un corpus dont les documents contiennent de l'information concernant différents domaines (par exemple sport, culture, politique).

Chaque aspect de l'hétérogénéité d'un corpus est qualifié par un degré 0 pour homogène ou 1 pour hétérogène. L'hétérogénéité du corpus est alors définie comme un triplet (structurelle, physique, sémantique). Par exemple, le corpus journal est de degré d'hétérogénéité (1, 0, 1), ce qui signifie que le corpus est de structure hétérogène, de support homogène (textuel) et de domaine hétérogène.

b) Le volume

Le volume d'informations contenues dans un corpus s'évalue en nombre de documents, en nombre de pages ou en nombre d'octets. Par hypothèse, une page est équivalente à 1,5 kilo octets. Par exemple le Web fournit un corpus de volume de un milliard de pages, équivalent à 150 Giga octets [Bar00]. Une bibliothèque comporte une dizaine de milliers de documents.

c) L'évolution

L'évolution d'un corpus définit le pourcentage de documents modifiés durant un intervalle de temps. Par exemple une étude récente menée aux E.U fait état d'une évolution du Web de l'ordre de 40% des pages référencées en un mois. Le corpus des bibliothèques est beaucoup plus stable (de l'ordre de 1%), car les opérations sur ce corpus concernent exclusivement l'ajout et la suppression en cas de perte de documents et sont donc moins fréquentes. Cette caractéristique est importante dans le cas du Web. En effet, la fréquence de mise-à-jour de la base d'index est de l'ordre de 4 à 6 semaines pour les outils de recherche en texte intégral [And00]. Or puisque les pages Web et leur emplacement sont modifiés plus fréquemment, la base d'index a une image non représentative de l'existant, provoquant des erreurs, par exemple une référence à une page qui n'existe plus est donnée en réponse.

d) Le type du corpus

Le type de corpus est défini par des contraintes sur l'ensemble des éléments de ce corpus. Un corpus est de type intranet, internet, Web, ou collection spécialisée.

Suivant le type du corpus un identifiant est attribué au document. Par exemple, un livre de bibliothèque possède un numéro ISBN, tandis qu'une page sur le Web est identifiée par une Uniform Resource Location (URL). Ce qui nous amène à énoncer l'hypothèse suivante : un document a toujours un identifiant et cet identifiant est unique.

3.2.3 Le gestionnaire d'index

Le gestionnaire d'index (1) est responsable de la création des informations nécessaires au gestionnaire de requêtes, c'est-à-dire les représentants et la base d'index. La création des représentants constitue la phase de pré-traitement de l'information et inclut le cas échéant les processus de clusterisation et de classification du corpus.

a) Les représentants des documents

Le représentant d'un document est une description de son contenu ainsi que de ses caractéristiques (description externe, structure) sous une forme interprétable par le gestionnaire de requête. Ces caractéristiques sont explicitées par des méta-informations que nous avons définies dans le chapitre 2. Notons que le représentant d'un document peut être le document entier.

Les représentants sont caractérisés par :

La méthode de construction Un représentant peut être construit automatiquement ou manuellement.

Citons des méthodes de construction automatique comme la clusterisation (extraction automatique d'un représentant identique pour un ensemble de documents appartenant au même cluster), la classification automatique, la création d'un représentant par lemmatisation et pondération par tf*idf (term frequency * inverted document frequency).

Une méthode manuelle consiste à fournir une description d'un document par un humain. Par exemple, dans les bibliothèques, le représentant nommé notice documentaire est construit par un documentaliste.

Les méta-informations Les méta-informations sont un ensemble d'informations semi-structurées qui sont normalisées ou non. Un exemple bien connu de normalisation est la spécification de Dublin Core Metadata [WGMD95]. Les méta-informations sont associées aux représentants par différentes relations. Soit les méta-informations ne sont pas utilisées pour construire les représentants, soit elles sont exactement les représentants, soit elles sont utilisées pour décrire les représentants. Les représentants ont avec les méta-informations une relation d'indentité (les méta-informations sont les représentants), d'inclusion (les méta-informations sont une partie des représentants) ou de description (les méta-informations décrivent les représentants).

Le format Un format définit les règles de syntaxe, de représentation d'un document. Par exemple, les formats XML sur le Web, MARC pour les bibliothèques, SOIF pour un système de recherche particulier.

La structure La structure est textuelle (linéaire) ou complexe (hiérarchisée). Une structure est linéaire lorsque le texte constitue une suite de caractères sans distinction de parties de texte ni de champs décrivant un contenu informationnel. Au contraire, une structure est hiérarchisée lorsque le texte est composé de parties imbriquées et/ou lorsqu'il est décrit par des éléments structurés. Une notice bibliographique est un représentant structuré d'un élément d'un corpus de documents de type bibliographique. La recopie du contenu textuel d'un document est un représentant non structuré, par exemple le contenu textuel d'une page Web en excluant les balises.

La granularité La granularité est définie par l'unité de transfert qui transite entre deux modules. La granularité peut être la page. Ce degré est déterminé à partir de la plus petite unité d'information que peut représenter un document. Par exemple ce peut être la page Web, ou une ligne, voire un terme. Cette caractéristique est très importante, car elle a un impact sur l'indexation, la recherche et la présentation des résultats. Par exemple pour la majeure partie des outils de recherche sur le Web, la granularité est fixe, c'est la page HTML physique. En revanche, d'autres systèmes proposent une granularité variable, comme le système HyPursuit.

b) L'index

Il s'agit de la construction de la base d'index à partir des représentants de documents. Dans notre acception du terme, l'indexation consiste à créer un chemin d'accès entre le ou les termes d'indexation et le document associé. Les termes d'indexation sont choisis à partir des représentants des documents, mais peuvent être complétés par d'autres données au moment de l'indexation, par exemple par la prise en compte de la structure de l'information.

L'index ou base d'index contient donc entre autres les termes d'indexation associés aux documents, les chemins d'accès aux documents et leurs références respectives.

Une base d'index est caractérisée par :

La structure ou implémentation de l'index Il s'agit de représenter un index en utilisant une structure de donnée adéquate, adpatée aux besoins. Il existe différentes structures d'index dont la plus répandue car la plus facile à mettre en œuvre est le fichier inversé.

Le stockage Il existe différentes manières de stocker un index, afin d'optimiser la phase de recherche tout en prenant en compte le volume de l'index. Pour stocker des index liés à des données structurées, il est préférable d'utiliser une structure externe comme les bases de données relationnelles, ou les bases de données objets. Lorsque les données n'ont pas de structure particulière, comme c'est souvent le cas dans le domaine de la recherche d'information, l'index est stocké dans un système de fichiers ou dans une structure particulière, par exemple dans un B-tree, une liste chaînée, une table de hashage.

Le volume Selon la représentation du document, la structure et le moyen de stockage utilisés, le volume de l'index change. Par exemple, dans l'index du système Harvest que nous décrivons en section 3 de ce chapitre, la taille de l'index d'un document peut atteindre le double de la taille de son représentant.

Le langage d'indexation Le langage d'indexation est utilisé à la fois dans la construction des représentants de documents et par dérivation dans la construction de l'index. Un langage d'indexation est normalisé ou non. Par exemple un thésaurus est vu comme un langage d'indexation normalisé, tout comme une liste ordonnée de vocabulaire contrôlé.

L'architecture Un index peut être centralisé, c'est-à-dire stocké sur une seule machine, ou distribué, c'est-à-dire réparti sur plusieurs machines, comme c'est le cas pour le système Harvest.

La coopération Un SRI est coopérant lorsque plusieurs SRI concourrent en dialoguant entre eux pour obtenir la résolution d'un problème. Ici, il s'agit de la coopération des systèmes d'index pour la construction d'une base d'index.

3.2.4 L'interface utilisateur

Dans un système de recherche d'informations, le besoin d'informations des utilisateurs est exprimé à travers une interface utilisateur sous la forme d'une requête. Il s'agit de la phase d'interaction entre l'utilisateur et le système. L'interface utilisateur est caractérisée par :

a) Le langage de requête

L'utilisateur traduit son besoin d'information sous la forme d'une requête exprimée en langage naturel ou sous la forme d'un ensemble de mots clés connectés par des opérateurs booléens. Une requête peut être exprimée dans un langage structuré (SQL), et être facilitée par l'aide d'un thésaurus.

Dans des systèmes multimédia plus sophistiqués, la requête peut être beaucoup plus complexe. Par exemple dans les systèmes [Fa97], [SF97], l'interface utilisateur permet d'exprimer des requêtes comportant des attributs aussi variés que la couleur, la texture, la forme, le mouvement de l'objet, une image existante, ou bien permet de spécifier des régions avec leur couleur.

b) La présentation des résultats

Une fois que les représentants des documents pertinents à la requête sont retrouvés par le gestionnaire de requêtes, l'interface utilisateur se charge de présenter les résultats à l'utilisateur. La présentation la plus utilisée est une liste de références aux documents.

3.2.5 Le gestionnaire des requêtes

La requête est envoyée au gestionnaire de requête. Celui-ci est caractérisé par la compréhension de la requête et la fonction de correspondance.

a) La compréhension de la requête

Le gestionnaire de requêtes traduit la requête selon un modèle de requête interne au système, afin d'exprimer le besoin de l'utilisateur dans un langage similaire à celui utilisé dans les index.

Le moteur de recherche met en correspondance le modèle de requête et la base d'index, afin de juger si un document référencé dans cette base d'index est pertinent ou non par rapport à la requête. Pour cela il procède par filtrage et éventuellement classement grâce à une fonction de correspondance.

b) La fonction de correspondance

La fonction de correspondance, appelée aussi fonction d'appariement, est déterminée par différents modèles. Par exemple le modèle booléen, le modèle vectoriel et le modèle probabiliste (voir en annexe A). À partir de cette fonction de correspondance, il est possible d'obtenir un classement des réponses, les documents les plus pertinents étant placés en tête.

3.2.6 Types d'architectures fonctionnelles

Les SRI sont classés selon un type d'architecture déterminé en fonction des critères de distribution, d'autonomie, et d'hétérogénéité (figure 3.3). Cette classification est inspirée des modèles d'architectures de bases de données distribuées dans [OV91].

L'autonomie se réfère à la capacité des différents composants du système à être totalement indépendants dans le traitement de la requête et à ne pas s'échanger d'informations. Un système totalement autonome est un système qui ne prend pas en compte les systèmes existants.

La distribution concerne les données. Soit les données sont physiquement réparties sur plusieurs sites et des mécanismes de communication entre les sites existent, soit les données sont sur un seul site. Un système centralisé n'est pas scalable.

L'hétérogénéité est fonction de trois paramètres : la structure, la sémantique, le support. La non prise en compte de l'hétérogénéité est un facteur de bruit et de silence.

A partir de ces trois critères, nous définissons les types d'architectures suivantes : architecture centralisée (sur le contrôle) homogène ou hétérogène, architecture fédérée homogène ou hétérogène, architecture distribuée homogène

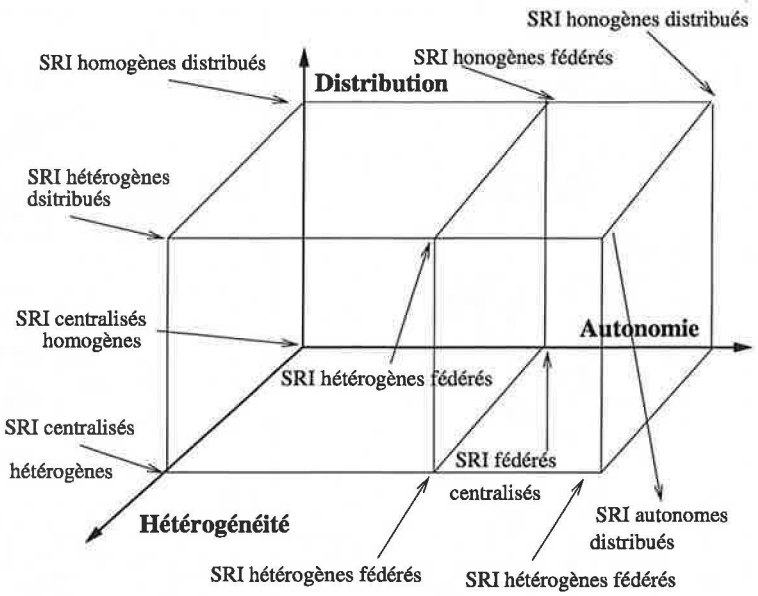


FIG. 3.3 – Types d'architectures de SRI

ou hétérogène.

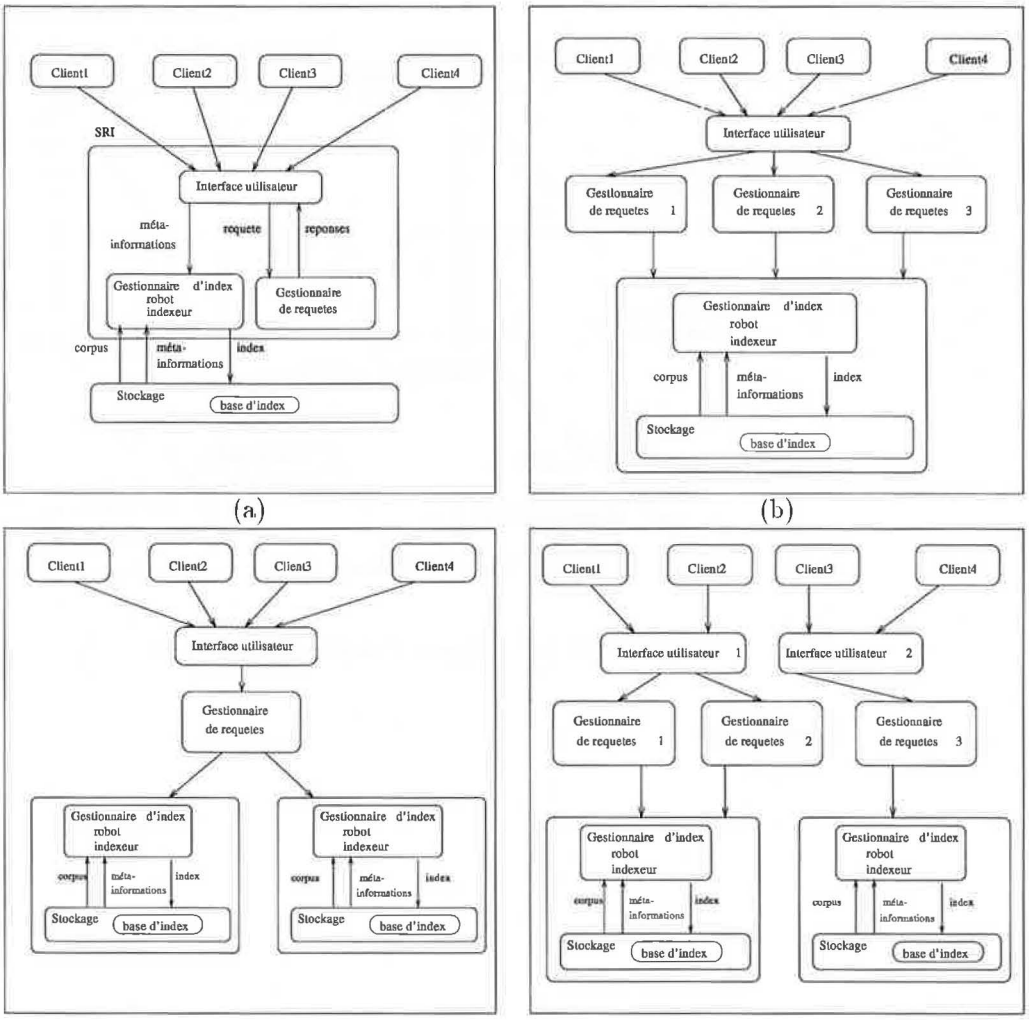


FIG. 3.4 – Les différentes architectures des SRI

Dans l'architecture centralisée figure fig.1, le gestionnaire d'index, la base d'index et la recherche sont centralisés. Ce type d'architecture est fréquemment utilisé parmi les outils de recherche sur le Web, car plus simple à mettre en œuvre.

Lorsque l'index est recopié sur plusieurs machines et que la recherche s'effectue sur chaque machine indépendamment des autres index, l'architecture est qualifiée de centralisée et redondante. Ce type d'architecture a pour but de dupliquer une architecture totalement centralisée afin d'augmenter la performance globale mais sans qu'il y ait ni distribution ni échange d'informations entre les serveurs d'index. Cette architecture est coûteuse en terme d'espace de stockage.

Dans l'architecture figure fig.2, l'indexation est distribuée et la recherche est centralisée avec plusieurs clients.

Dans l'architecture figure fig.3, l'indexation est centralisée et la recherche est parallèle. Cette architecture est scalable au niveau de l'indexation et la recherche parallèle accélère les traitements. Dans le cas d'une recherche parallèle, les problèmes de redondance de l'information apparaissent, ce qui est un facteur de bruit.

Dans l'architecture figure fig.4, l'indexation est distribuée et la recherche est distribuée. La recherche distribuée permet une communication entre les composants.

3.3 Les systèmes de recherche d'informations sur le Web

Nous présentons un ensemble de systèmes de recherche représentatifs sur le Web : les moteurs de recherche, comme AltaVista, Google et NorthernLigth, les annuaires comme Yahoo !, les outils spécifiques comme Ontobroker, Aliweb, SHOE et les systèmes distribués comme Harvest, HyPursuit ou InfoSleuth. Cette étude a pour but de dégager les points importants ainsi que les limites de l'existant pour comprendre notre démarche et la justifier.

3.3.1 Alta Vista

Alta Vista est un moteur de recherche en texte intégral.

Développé en 1995 dans les laboratoires Digital à Palo Alto (Californie), Alta Vista se compose d'un robot, Scooter qui collecte les pages (10 millions par jour

en 1998), d'un serveur Web, d'un serveur d'index et d'une interface utilisateur.

En 1998, sont apparus l'affinement de requêtes (en posant des nouvelles questions en langage naturel) avec Askjeeves, le filtre sur les composants multimédia. En 2000, Raging search personnalise un profil d'utilisateur afin de filtrer sur un critère comme la langue, un domaine d'intérêt.

a) Représentation des donnéss

Alta Vista considère uniquement le niveau de représentation physique, c'est-à-dire qu'un document est représenté par une page du Web, unité physique de stockage. Alta Vista ne gère ni l'indépendance physique ni l'indépendance logique.

b) Architecture fonctionnelle

L'architecture fonctionnelle est schématisée figure 3.5. Les chiffres sont issus de http ://www.abondance.fr. AltaVista répond au type d'architecture centralisé et hétérogène, avec redondance des composants.

Lorsqu'un client Web se connecte à l'un des trois serveurs qui affichent l'interface utilisateur, le traitement d'une requête se déroule comme suit :

Saisie de la requête par un utilisateur via le client Web. Traduction de la requête par l'interface utilisateur et envoi de la requête interne à l'un des sept serveurs qui hébergent les gestionnaires de requêtes. Chacun de ces serveurs contient une copie de la base d'index centralisée construite par le gestionnaire d'index. Interrogation de la base d'index recopiée par le gestionnaire de requête via la fonction de correspondance. Envoi des résultats à l'interface utilisateur qui se charge de les conserver et de les transmettre au client par paquets de pages.

Algorithme correspondant :

```
declare-var
  IU:UserInterface
  GI:IndexManager
  GR:RequestManager
  E:Evaluation
  R:Result
input : query with n terms
```

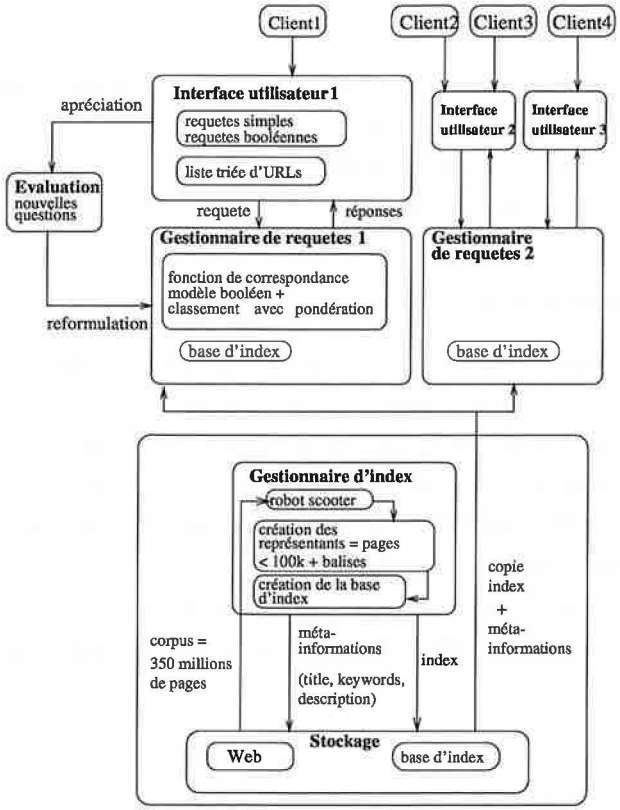


FIG. 3.5 – Architecture fonctionnelle d'Alta Vista

```
output : list of URLs
begin
IU.read(query)
IU.sendQuery(GR,query)
GR.evaluate(query)
GI = GR.select(IndexManager)
Index = GI.requestIndex()
R = GR.matchingFunction(query, Index)
print R
end
```

c) Le corpus

En septembre 2000, Alta Vista collecte 350 millions de pages Web. Le corpus du Web est de degré d'hétérogénéité maximum (1,1,1) (p.41). Les pages Web sont identifiées de manière unique par leur URL. Les pages Web sont collectées par le robot scooter qui stocke les URLs des pages sur son disque de 30 Go.

d) Le gestionnaire d'index

Alta Vista conserve le contenu du corps du texte des pages Web jusque 100 Koctets. Au delà, il y a perte d'informations. Les représentants sont extraits automatiquement des pages. Les ressources telles que les images sont indexées selon le nom du fichier, par exemple le mot-clé retenu pour le fichier voile.gif est voile. L'attribut ALT de la balise IMG est aussi pris en compte. Enfin, l'intitulé de l'URL est également indexé.

Les méta-informations sont déduites des balises META description et META keywords, ainsi que de la balise title. Par exemple, dans la page HTML suivante :

```
<HTML>
<HEAD>
<TITLE> La pollution de l'eau</TITLE>
<META name = ''keywords''  content = ''pollution, eau''></META>
<META name = ''description''  content = ''la pollution de l'eau
est un problème de nuisance géré par des organismes specialisés comme
l'ADEME.''></META>
</HEAD>
<HTML>
```

Ces informations sont conservées dans la base d'index, les champs indexés sont représentés dans le tableau 3.1.

Alta Vista définit une importance relative entre les balises des représentants par ordre décroissant le titre, le corps du texte et les balises META. La granularité de l'information est fixe, celle de la page.

Tous les mots des représentants sans exceptions sont indexés. Deux index coexistent, le premier associe à chaque terme d'index l'URL des documents qui le contiennent, le deuxième stocke pour chaque terme indexé son poids (fréquence d'occurrence du terme dans le document) ainsi que sa position absolue dans le document.

TAB. 3.1 – Champs indexés dans Alta Vista

Champs indexés	exemple	taille maximale	pondération
TITLE	La pollution de l'eau	100 caractères	0.3
KEYWORDS	pollution	1024 caractères	0.1
DESCRIPTION	la pollution de l'eau est un problème de nuisance géré par des organismes specialisés comme l'ADEME	1024 caractères	0.1
URL	http ://www.organisme-env.fr/eau	—	0
BODY	corps du texte	100 Koctets	0.2

Le processus d'indexation est centralisé sur une machine dont la capacité du disque est 180 Go.

f) L'interface utilisateur

Deux interfaces sont disponibles, sous la forme de recherche simple et recherche avancée. A priori la recherche simple fait intervenir le modèle vectoriel alors que la recherche avancée est booléenne avec la possibilité d'interroger l'index en utilisant les opérateurs OR, AND, NOT, NEAR, le parenthésage ainsi qu'une interrogation sur les balises *title, meta, author, anchor et sur les champs url, domain.*

Les résultats sont présentés sous la forme d'une liste triée de documents dont les champs suivants sont affichés : le titre, la description du champ META, l'URL.

g) Le gestionnaire de requêtes

La fonction de correspondance prend en compte les paramètres pré-cités afin d'évaluer la pertinence d'un document face à une requête et de proposer un classement dans les pages données en réponse. Par exemple les termes trouvés en début du document auront plus de poids, la balise title a un poids plus fort.

h) Conclusion

Alta Vista est très représentatif des outils de recherche en texte intégral, au sens où la majorité des outils sur le Web suivent le même schéma de collecte, de construction d'index et ont une interface d'interrogation booléenne. Ce

que nous pouvons reprocher à ces outils, est qu'ils ne prennent pas en compte dans leur corpus et dans le gestionnaire d'index ni la sémantique ni la structure des documents. Dans la phase de collecte du corpus et de la création des représentants, les balises META, TITLE sont extraites. La prise en compte de la méta-information dans le format HTML, a pour but de mieux cibler le contenu sémantique de chaque page. Cependant nous notons que l'utilisation de mots-clés libres ne résoud pas l'ambiguité du vocabulaire non contrôlé et que ni le contexte ni la structure ne sont explicités. En effet, les mots-clés sont locaux à une page et ne propagent pas le contexte, ici environnement n'apparaît pas dans la page car il a déjà été intoduit dans la page d'accueil du site. Il y aura du silence avec la requête "environnement AND pollution de l'eau" spécifiant le contexte environnement.

Ces outils ne résolvent pas le problème du bruit et du silence. La présentation des réponses sous la forme d'une liste de référence de pages Web avec un extrait issu des premières lignes de la page, dénuée de contexte est difficile à exploiter. La prise en compte de la balise META keywords peut être source d'erreur, à partir du moment où des utilisateurs l'utilisent afin d'être classés en tête de liste et non pour décrire leur site.

Dans la présentation des résultats, le contexte associé à une page n'apparaît pas, ni la structure. Le champ description est celui fourni par la balise META description et s'il n'est pas renseigné, il est remplacé par les premières lignes de la page Web. Il y a un manque d'homogénéité dans les réponses.

Enfin la centralisation de l'index pose le problème de scalabilité et est une des causes à l'origine du bruit et du silence.

Parmi les outils en texte intégral qui sont apparus récemment et qui ont obtenu du succès nous pouvons citer Hotbot (en 1996), Google (en 1998), dont la mesure de pertinence est calculée grâce à la popularité d'un site, (l'indice de popularité d'une page est fonction du nombre de pages qui référencent cette page). D'autres outils, comme Excite (1995) essaient de regrouper les pages en réponse par thème à postériori, ou comme HyPursuit (1996) a priori. Notons que dans le cas de HyPursuit, le regroupement de pages s'effectue par calcul de similarité en fonction des contenus des pages et des liens hypertextes. NorthernLigth (1997) a la particularité de regrouper a posteriori des réponses selon des catégories déterminées à partir de la requête. Cependant, ces groupements sont tous basés sur le contenu des pages, et ne prennent en compte ni les méta-informations ni la structure globale des sites, ni le contexte, par conséquent ils ne donnent pas de réponses satisfaisantes par rapport au bruit et au silence.

3.3.2 Yahoo!

Né en 1994, Yahoo! est un outil de recherche thématique, proposant une arborescence universelle de thèmes. Chaque thème correspond à une catégorie dans l'arbre et en réalité, une catégorie peut appartenir à plusieurs branches de l'arbre. Yahoo! construit une base unique contenant la liste des URLs connues sur tous les sites déclarés. Cette liste est construite par ajout d'URL par l'utilisateur. Chaque page est indexée manuellement dans une ou plusieurs catégories, grâce au formulaire de description de la page rempli par son auteur.

a) Représentation des données

Yahoo!, tout comme Alta Vista considère uniquement le niveau de représentation physique, qui est la page. Cependant, contrairement à Alta Vista, Yahoo! propose une construction manuelle et contrôlée par l'auteur d'un site. Yahoo! ne gère ni l'indépendance physique ni l'indépendance logique.

b) Architecture fonctionnelle

L'architecture fonctionnelle est schématisée figure 3.6. Les chiffres sont issus de `http://www.abondance.fr`. L'architecture de Yahoo! est centralisée, autonome et hétérogène.

Le fonctionnement est le suivant :

```
Soumission d'une URL et des méta-informations par l'utilisateur par
l'intermédiaire d'un formulaire de saisie.
Création du représentant qui est l'ensemble des méta-informations.
Indexation des méta-informations.
Interrogation booléenne sur l'index des méta-informations.
Possibilité d'interroger sur un sous-ensemble des catégories.
```

c) Le corpus

En septembre 2000, le corpus de Yahoo se compose de 1.8 millions de représentants de sites Web. Chaque élément est un représentant d'un site Web, comportant une fiche descriptive du site et l'URL de la page d'accueil du site. Le corpus est constitué manuellement, grâce à un formulaire d'enregistrement qui permet de ranger une page dans une arborescence universelle de thèmes.

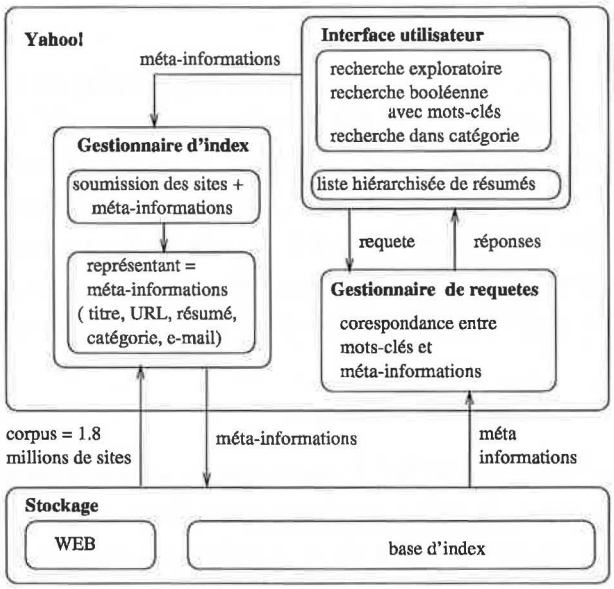


FIG. 3.6 – Architecture fonctionnelle de Yahoo

d) Le gestionnaire d'index

Les représentants des sites sont construits manuellement. Les méta-informations sont les représentants et sont renseignées par l'auteur. Il s'agit des titre, auteur, résumé, mots-clés et choix de placement dans l'arborescence (catégorie). Ces informations sont conservées dans la base d'index, les champs indexés sont représentés dans le tableau 3.2.

TAB. 3.2 – Champs indexés dans Yahoo!

Champs indexés	exemple
TITLE	La pollution de l'eau
CATEGORY	business/environment/pollution
DESCRIPTION	la pollution de l'eau est un problème de nuisance géré par des organismes specialisés comme l'ADEME
URL	http ://www.organisme-env.fr/eau
E-MAIL	adresse électronique de l'auteur

e) L'interface utilisateur

Yahoo ! propose un guide hiérarchique par thème, donc permet une recherche exploratoire. Les thèmes principaux regroupent les grandes classifications (littérature, art, sciences...). La recherche peut également s'effectuer par mots-clés, sur une chaîne ou une sous-chaîne de caractères. L'utilisateur peut interroger la base d'index fabriquée à partir des informations fournies par les auteurs des pages, soit dans l'arborescence générale, soit dans le sous-arbre courant (à partir du nœud où l'on se trouve).

f) Le gestionnaire de requêtes

La recherche peut être filtrée sur les champs : titre, URL, résumé.

g) conclusion

Yahoo ! est représentatif des annuaires thématiques, qui offrent une solution au problème de bruit mais augmentent le silence par un taux de couverture faible, de l'ordre de 0.2% du Web comparé aux plus gros indexeurs qui ont un taux de couverture estimé à 20% du Web.

Les réponses sont cependant plus pertinentes car elles affichent un résumé significatif, plutôt que les premières lignes d'une page, qui ne permettent pas de juger de la pertinence de la réponse. Nous constatons donc que l'usage des méta-informations est un facteur de meilleure représentativité des sites et la classification hiérarchisée des sites selon des catégories pré-définies un moyen de prendre en compte le contexte dans lequel se situe une information.

Citons un autre exemple d'annuaire de site, AliWeb, qui lui n'est pas thématique. Né en 1995, Aliweb est un système qui propose une autre approche par rapport aux outils de recherche classique. En effet, plutôt que d'indexer les pages telles quelles, il s'agit de proposer d'ajouter à la racine d'un site un fichier de méta-informations décrivant ce site, afin que ce fichier puisse être retrouvé et interprété par l'outil de recherche. Le moteur de recherche Aliweb utilise le formalisme IAFA pour décrire et indexer certaines pages du site plutôt que toutes les pages. Avec cette méthode, les services et les pages clés sont décrites mais l'organisation du site n'est pas décrite et les contenus des pages ne sont pas indexées.
Il y a donc une solution au problème du bruit mais pas du silence.

Cependant, cette prise en compte de la sémantique et du contexte est limitée

au niveau de représentation physique, seule représentation de document traitée par les outils actuels. De plus, toutes les pages d'un site ne sont pas indexées, et il manque la prise en compte du niveau de représentation logique de ces pages, qui permettrait de mieux expliciter la sémantique d'un site ainsi que le contexte local aux pages d'un même site. Il existe de nombreux annuaires thématiques, comme Voilà, en France, apparu en 1996, géographiques ou régionaux, c'est-à-dire se limitant à la couverture d'un domaine géographique plus restreint. Cette tendance correspond à un besoin réel de cibler des recherches par filtrage sur une catégorie mais aussi sur une culture, une langue ou une région précise. Notons aussi que la plupart des outils thématiques intègrent un moteur de recherche en texte intégral et vice-versa. Il y a une tendance à l'exhaustivité et à la complémentarité des services offerts par chaque outil.

3.3.3 MetaCrawler

MetaCrawler est un méta-chercheur, au sens où il fournit un service de recherche en parallèle en interrogeant plusieurs moteurs de recherche. Disponible à l'université de Washington depuis 1995, il fournit aux utilisateurs une interface unique pour interroger plusieurs moteurs différents, tels que Lycos, Alta Vista et fournit d'autre fonctionnalités pour fusionner les résultats. MetaCrawler n'est pas réellement un SRI, puisqu'il ne dispose que d'une interface utilisateur.

a) Représentation des données

Pas d'indépendance physique ni logique.

b) Architecture fonctionnelle

L'architecture fonctionnelle de MetaCrawler est schématisée figure 3.7. L'architecture de MetaCrawler est distribuée et parallèle, autonome et hétérogène.

Le fonctionnement est le suivant :

```
Réception d'une requête utilisateur à travers une interface centralisée.
Normalisation de la requête pour chacun des SRI à interroger.
Envoi de la requête à chacun de ces SRI en parallèle.
Récupération des résultats et agrégation de ceux-ci.
Suppression des redondances.
Présentation des résulats sous la forme d'une liste triée selon une mesure
```

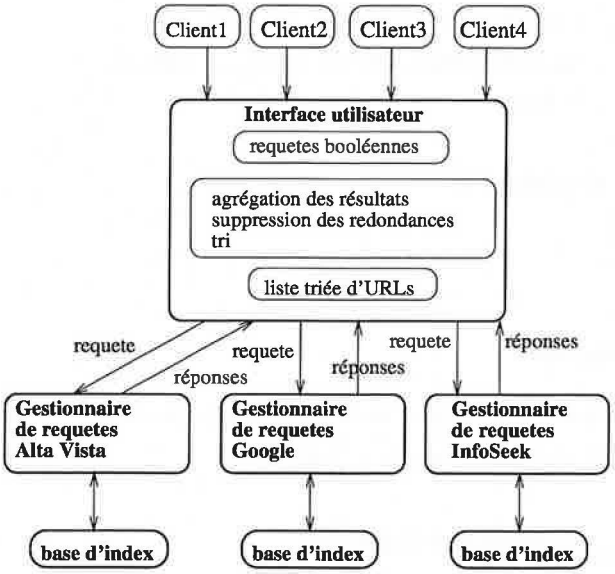


FIG. 3.7 – Architecture fonctionnelle de MetaCrawler

de pertinence globale.

Algorithme correspondant :

```
declare-var
  IU:UserInterface
  GI:IndexManager
  GR:RequestManager
  E:Evaluation
  Resp:Response
  S : set of Response
  R:Result
  G: set of RequestManager
input : query with n terms
output : list of URLs
begin
IU.evaluate(query)
for each GR in G do
  IU.sendQuery(GR,query)
  GI = GR.select(IndexManager)
```

```
  Index = GI.requestIndex()
  Resp = GR.matchingFunction(query, Index)
  S  = S + Resp
end for
R = IU.mergeResult(S)
print R
end
```

c) Le corpus

Le corpus est constitué de la réunion des corpus de chaque outil de recherche participant au processus de recherche.

d) Le gestionnaire d'index

MetaCrawler ne possède pas de propre gestionnaire d'index puisqu'il se sert des index des outils de rcherche qu'il interroge.

e) Le gestionnaire de requêtes

L'utilisateur entre une requête sous la forme d'une liste de mots-clé avec la possibilité de rechercher soit tous les mots, soit un des mots ou une phrase complète. MetaCrawler prend cette requête et la formate correctement pour chaque service de recherche. Ensuite il doit agréger les résultats en éliminant les doublons. Il est difficile de trouver un critère de classement car chaque outil a sa propre méthode pour calculer les scores.

3.3.4 Ontobroker

Ontobroker [FAD+99] est un prototype de recherche développé en 1998, au laboratoire AIFB à l'université de Karlsruhe en Allemagne. Ontobroker repose sur un formalisme d'ontologie basé sur la logique de Frame [KLW95], destiné à générer et inférer des méta-informations sur le Web. Ce formalisme est utilisé pour annoter des documents. Il comprend les primitives de sous-classe, d'instance, de déclaration d'attributs, de valeurs d'attributs et de relations exprimées à l'aide de la logique de prédicats. Ce formalisme est défini en RDF (Resource Description Framework) [W3C], format récemment adopté par le W3C pour décrire des ontologies, des graphes conceptuels.

a) Représentation des données

Dans la mesure où Ontobroker offre un langage d'annotations de pages, et structure le contenu de chaque page grâce à un modèle logique, il y a indépendance physique des données.

b) Architecture fonctionnelle

L'architecture d'Ontobroker est centralisée, autonome et hétérogène. L'hétérogénéité des données est prise en compte par l'ajout des méta-informations dans les pages HTML.

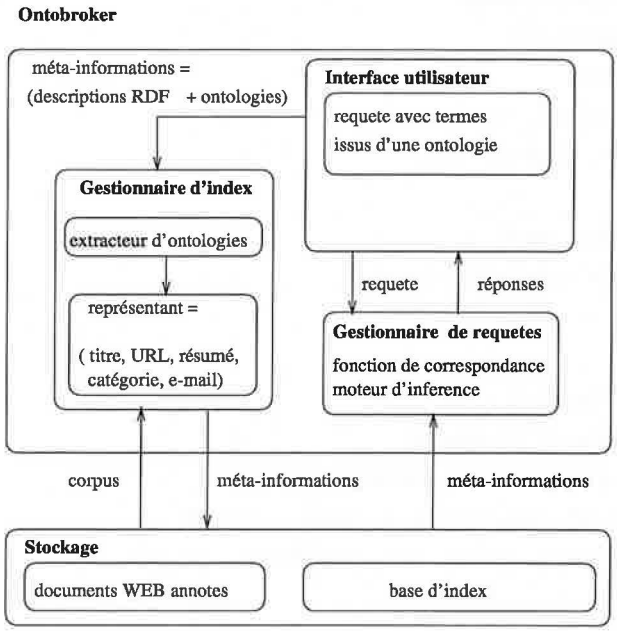


FIG. 3.8 – Architecture fonctionnelle d'OntoBroker

Le déroulement d'une requête est le suivant :

Expression de la requête grâce à une interface permettant de parcourir l'ontologie et de sélectionner les termes qui composent la requête.
Evaluation et traduction de la requête à l'aide du formalisme de requête basé sur la logique de Frame.

```
Invocation de la fonction de correspondance (via le moteur d'inférence)
par le gestionnaire de requête qui utilise les faits
et règles provenant de la base de connaissance afin de déterminer les
réponses qui sont des objets stockés dans la base de données.
```

Et voici l'algorithme correspondant :

```
declare-var
  O:Ontology
  t:string
  IU:UserInterface
  GI:IndexManager
  GR:RequestManager
  E:Evaluation
  R:Result
input : query with n terms
output : list of URLs
begin
while (not end)
{ t = IU.select(O.terms())
  query = query + t
}
IU.read(query)
IU.sendQuery(GR,query)
queryform = GR.evaluate(query)
Index = GI.requestIndex(facts, rules)
R = GR.matchingFunction(queryform, Index)
print R
end
```

c) Le corpus

Ontobroker est adapté pour traiter des corpus multimédia, puisqu'il fournit un moyen de les décrire grâce aux ontologies [SK98]. Le corpus est le Web.

d) Le gestionnaire d'index

Les représentants sont construits soit via un extracteur de connaissances à partir des pages HTML soit ce sont les annotations des documents créées par

les utilisateurs à l'aide d'une ontologie. Voici un exemple de document annoté (encodé dans HTML) :

```
<html>
<head><TITLE> Richard Benjamins </TITLE>
<a ONTO="page:Researcher"> </a>
</head>

<H1> <A HREF="pictures/id-rich.gif">
<IMG  align=middle SRC="pictures/richard.gif"></A>
<a ONTO="page[photo=href]"
HREF="http://www.iiia.csic.es/~richard/pictures/richard.gif" ></a>
<a ONTO="page[firstName=body]">Richard</a>
<a ONTO="page[lastName=body]">Benjamins </a>
</h1> <p>

<A ONTO="page[affiliation=body]" HREF="#card">
Artificial Intelligence Research Institute (IIIA)</A> -
<a href="http://www.csic.es/">CSIC</a>, Barcelona, Spain <br>
```

Cet exemple illustre les primitives d'annotations : un objet identifié par un URL est une instance d'une classe, une relation entre deux ou plusieurs objets peut être établie.

L'index, tableau 3.3 comporte l'ontologie (hiérarchie de concepts, attributs, règles) et les instances de l'ontologie qui sont les représentants.

TAB. 3.3 – Index d'Ontobroker

Données indexés	exemple
Hiérarchie de concepts	object Person : :object Researcher : :Person
Attributs	Person[firstName, lastname, email]
Règle	FORALL Person1 p1, Person2 p2 p1 :Researcher[worksWith $->$ p2] $<--$ p2 :Researcher[worksWith $->$ p1]

e) L'interface utilisateur

L'interface utilisateur permet une formulation des requêtes en parcourant une ontologie et en sélectionnant les termes pour constituer la requête.

f) Le gestionnaire de requêtes

Le gestionnaire de requêtes utilise la logique de Frame pour inférer des connaissances qui l'aideront à déterminer quelles réponses, sous forme d'objets (représentants de documents) sont pertinents à une requête. Pour trouver la page HTML, le nom de famille et l'adresse email de tous les chercheurs qui se prénomment Richard, il faut réaliser la requête suivante :

```
FORALL Obj, LN, EM <-
  Obj:Researcher[firstName->>''Richard''; lastname->>LN;email->>EM]
```

g) conclusion

Grâce à l'introduction d'un formalisme basé sur la logique et un mécanisme d'annotations de pages HTML, Ontobroker a montré la faisabilité d'inférer des connaissances en incluant des méta-informations directement sur les pages. Ontobroker permet ainsi de répondre à des requêtes structurées qui ne sont pas traitées par les autres moteurs. Par là, Ontobroker réduit le bruit et le silence.

SHOE [HHL99] est un système qui permet également aux auteurs d'annoter leurs pages HTML en utilisant une ontologie basée sur le contenu des pages. Des entités sont déclarées pour un ensemble de documents et des sous-sections de documents. SHOE gère l'indépendance physique. Les relations entre entités sont déclarées ainsi que les attributs. Les entités sont classifiées dans un schéma de classification. SHOE définit un modèle entité-relation au-dessus des pages Web. SHOE permet l'ajout de sémantique directement en HTML, par exemple :

```
<ONTOLOGY ''our-ontology'' VERSION=''1.0''>
<ONTDEF CATEGORY=''Person''>
<ONTDEF RELATION=''firstName''>
   ARGS=''Person STRING''>
</ONTOLOGY>
```

Les annotations se limitent au sein d'une même page et ne reflètent pas la description formelle d'un ensemble plus large de documents. Une autre limite est la restriction de l'interrogation aux seuls éléments du modèle, il manque une intégration de la recherche sur le contenu textuel de la page. De plus, peu d'explications sont données sur la constitution d'une ontologie commune ou au contraire sur la possibilité d'extension vers des ontologies plus spécifiques à des domaines de la connaissance, ainsi que la définition d'une infrastructure pour

gérer ces informations sur l'exemple de Warwick framework [LLD96]. Une autre difficulté de ces systèmes est la mise en œuvre relativement complexe et coûteuse des annotations sur chaque page HTML. De tels systèmes sont susceptibles de ne pas avoir une grande portée sur le Web, mais sont plutôt réservés à des corpus de type Intranet.

3.3.5 HyPursuit

HyPursuit [RW96] a été créé en 1996 au MIT, Etats-Unis. Sa particularité est d'agréger les documents hypertextuels en utilisant une fonction de similarité qui combine à la fois le contenu textuel ainsi que les liens hypertextes des documents. Des *fonctions d'abstraction* résument les clusters. Différents clusters peuvent coexister.

a) Représentation des données

Par la création des hiérarchies de clusters, HyPursuit offre une indépendance physique des données.

HyPursuit offre également une indépendance logique des données. Le fait que plusieurs hiérarchies de clusters co-existent signifie que plusieurs vues existent à partir d'une hiérarchie de clusters de base.

b) Architecture fonctionnelle

L'originalité d'HyPursuit réside dans la hiérarchisation des serveurs contenant des clusters construits automatiquement à l'aide de fonctions d'abstraction. L'architecture est autonome, coopérante et distribuée. HyPursuit est une hiérarchie de SRI prédéfinie, par exemple déterminée par la hiérarchie de DNS. Les SRI de plus bas niveau clusterisent les pages HTML et fabriquent à l'aide d'une fonction d'abstraction un résumé sous forme d'une liste de mots-clés de chacun des clusters. Ces résumés seront alors utilisés par d'autres SRI pour construire des clusters de plus haut niveau grâce à la fonction d'abstraction. Lorsqu'une requête arrive au serveur, le gestionnaire de requête choisit de router la requête en consultant les résumés afin de savoir quels serveurs fils interroger. Il fusionne alors les réponses et les présente sous la forme de clusters.

c) Le corpus

Le corpus traité est le Web, l'hétérogénéité des données n'est pas prise en compte.

3.3.6 Harvest

Harvest [BDH+95] est un système issu de l'université du Colorado, conçu par D.R. Hardy, M.F. Schwartz et D. Wessels en 1994. Harvest se compose d'un ensemble de sous-systèmes *Collecteur*, *Courtier*, *Cache*, et *Replicateur* destinés à rassembler et à distribuer de l'information indexée. Les collecteurs, nommés robots dans notre architecture sont destinés à extraire l'information des sites Web locaux, régulièrement, ceci afin de réduire la consommation en bande passante. Les documents extraits sont transformés en objets SOIF. Les gestionnaires d'index nommés courtiers s'occupent du mécanisme d'indexation, ils trouvent l'information à partir des collecteurs ou copient les index d'autres courtiers. Ils fournissent l'interface d'interrogation sur les données collectées. Ce système requiert la coordination de plusieurs serveurs Web. En effet, un courtier ne peut s'adresser qu'à des collecteurs ou à d'autres courtiers qu'il connait et qui sont en service. L'intérêt est d'échanger des objets SOIF afin de télécharger un moindre volume d'information. Les mécanismes de cache et de réplicants de courtiers permettent de gérer la scalabilité du système. Afin de connaître les courtiers existants, Harvest gère un courtier spécial qui joue le rôle d'annuaire centralisé. Actuellement une centaine d'applications utilisent le système Harvest dont certaines au CIA et à la NASA. Notons qu'un courtier cumule les fonctions d'indexation, il a le rôle de gestionnaire d'index dans ce cas, et les fonctions d'interrogation, il joue alors le rôle de gestionnaire d'index dans notre terminologie.

a) Représentation des données

L'unité physique du fichier est choisie comme document. Il n'y a pas d'indépendance physique ni logique.

b) Architecture fonctionnelle

Harvest est de type architecture distribuée, fédérée et hétérogène. Cette architecture est une solution au problème de scalabilité. Cependant, le problème

de bruit et de silence n'est pas résolu. En effet, l'indexation est effectuée localement mais sans distinction de domaine, ni de structure.

L'architecture fonctionnelle est schématisée figure 3.9.

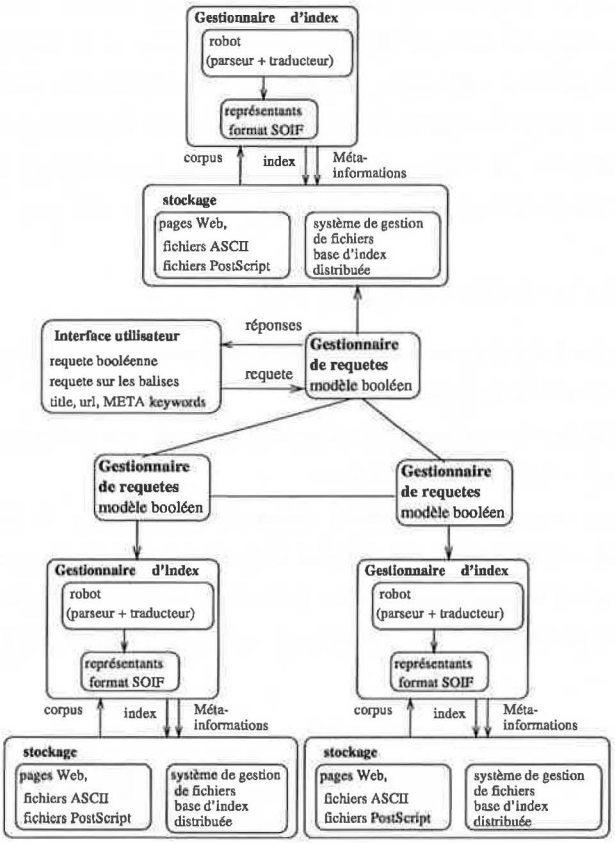


FIG. 3.9 – Architecture d'Harvest

Le corpus Le corpus est hétérogène (1,1,1), il comporte n'importe quelle page téléchargée à partir du Web, ou encore un fichier récupéré par le protocole ftp. Une caractéristique d'Harvest est sa capacité à traiter l'hétérogénéité physique. En effet, il prend en compte les fichiers accessibles sur le Web, tels que les documents postscripts, les documents formatés en LaTex, HTML, Framemaker, RTF, SGML, les documents images, les documents multimédia.

La collecte de document est automatique, grâce au robot appelé *Collecteur* qui se charge de collecter l'information soit en se plaçant sur le site fournisseur — la récupération des données est alors locale — soit en récupérant les informations

à partir de sites distants. Un fichier de configuration permet à l'administrateur de spécifier les sites, les pages de départ et le niveau de profondeur dans le parcours de ces pages.

Le but est de constituer un corpus de faible volume, contenant les pages de sites locaux, ceci afin de distribuer l'indexation et diminuer la consommation en bande passante. De plus, un corpus peu volumineux permet d'envisager des délais de rafraîchissement moindres et donc des réponses plus à jour.

Un élément du corpus est identifié par son représentant à l'aide d'un numéro d'identification unique en fonction d'un site, et fourni par le collecteur.

Les représentants des documents

Chaque document est analysé par un parseur, nommé résumeur. Le représentant est une description d'un document.

Le format du représentant propre à Harvest, est le format SOIF pour Standard Object Interchange Format.

Le représentant est construit automatiquement par le résumeur. Le résumé est établi à l'aide d'un traducteur des correspondances entre les attributs des documents d'origine (titre, date, auteur) et les attributs SOIF. Le tableau 3.4 donne un exemple de correspondance entre les attributs HTML et les attributs SOIF.

TAB. 3.4 – Correspondance des champs HTML et SOIF

$elementHTML$	$AttributsSOIF$
$< A >$	$keyword, parent$
$< A : HREF >$	$url - references$
$< BODY >$	$body$
$< CITE >$	$references$
$< H1 >$	$headings$
$< H2 >$	$headings$
$< H3 >$	$headings$
$< TITLE >$	$title$

Voici un exemple de représentant d'un document :

```
@FILE {http://agora21.emse.fr/partenaires/
Update-Time{9}: 895995647
Description{11}:        partenaires
```

```
Keywords{12}:    partenaires
Last-Modification-Time{9}:      894373388
Time-to-Live{7}:        2419200
Refresh-Rate{6}:        604800
Gatherer-Name{25}:      Example Gatherer Number 1
Gatherer-Host{17}:      groseille.emse.fr
Gatherer-Version{3}:    1.5
Type{4}:        HTML
File-Size{3}:   388
MD5{32}:        6f661a877632a39632645e3d74e752f9
auteur{14}:     Florent Breuil
body{142}:      COLS="100
ROWS="40,*" BORDER=0>
SRC="menu.html" NORESIZE
MARGINHEIGHT=0 SCROLLING="no" NAME="Fenetre2">
SRC="frame1.html"
NAME="Fenetre1">
title{11}:      partenaires
}
```

Le représentant d'un document est structuré par les balises SOIF, cependant le niveau de représentation logique du document n'est pas prise en compte.

L'index

Harvest pratique une indexation en texte intégral. Les attributs des représentants (attributs SOIF) correspondent aux champs indexés. Les représentants des documents sont implantés en fichiers inverses.

La base d'index est stockée dans un système de gestion de fichiers.

L'index est paramétrable, c'est-à-dire qu'il est possible de spécifier les informations qui doivent être indexées par l'intermédiaire d'un fichier de configuration. La taille de l'index varie en fonction de ces paramètres. Un index peut atteindre 2 à 3 fois la taille du document source.

Dans Harvest, l'indexation est distribuée.Deux types d'outils mettent en œuvre l'indexation :

- Le gestionnaire d'index fabrique à la fois la base d'index sur laquelle vont pouvoir se greffer d'autres serveurs, mais aussi l'outil qui va scruter des objets sur différents sites et maintenir les informations indexées. Le but

d'Harvest est de créer des gestionnaires dans des domaines spécifiques, afin d'éviter les problèmes de sémantiques liés à des index trop génériques.

- le gestionnaire d'index indexe l'information collectée par un ou plusieurs robots ou par d'autres gestionnaires d'index. Pour indexer une base, le gestionnaire de requête utilise une interface de requête dans un protocole bien défini, qui est un flot de données attribut-valeur. L'intérêt dans cette démarche est de faire apparaître une hiérarchie de "serveurs d'index", un gestionnaire d'index pouvant fournir de l'information à un ou plusieurs gestionnaires de requêtes. Les "vues" obtenues par une indexation en cascade sont un moyen de filtrer et de raffiner l'information. Cependant l'organisation globale des courtiers et des collecteurs n'est pas définie et il n'y a pas non plus de coopération entre ces entités.

La communication entre les courtiers et les collecteurs s'effectue via le protocole SOIF, les messages envoyés sont au format SOIF.

L'interface utilisateur

Les requêtes sont traitées par les *Courtiers.*

Harvest supporte les opérateurs booléens, les filtres sur des chaînes ou sous-chaînes de caractères, les expressions régulières. Les requêtes sont structurées sous la forme attribut-valeur ou combinées à l'aide d'attributs ou valeurs. Par exemple on peut avoir la requête : "Robby Naish" AND (Title :windsurfing)

Le gestionnaire des requêtes

Harvest utilise le modèle booléen comme type de fonction de correspondance. Grâce à l'opérateur *near*, il est tolérant aux erreurs (par exemple le terme "informatique" est équivalent au terme "infomatique").

3.3.7 Les systèmes multi-agents pour la recherche d'informations sur le Web

Une approche distribuée pour résoudre la recherche d'informations sur le Web concerne les applications issues des systèmes multi-agents. Nous distinguons parmi celles qui nous intéressent, deux types d'architectures d'agents. Les premières sont centrées sur le profil de l'utilisateur, pour construire des agents personnalisés ou des communautés d'agents avec les mêmes besoins et les mêmes intérêts, nous pouvons citer MINDS, [Mae94], [?], [HOY+99]. Les

deuxièmes sont orientées vers les ontologies qui représentent les connaissances que les agents possèdent sur eux-mêmes et sur leur environnement, citons les travaux de OntoBroker, Carnot, Vasant Honovar.

C'est sur ces dernières que nous allons porter notre étude, car elles serviront par la suite de base de comparaison avec notre système. Nous décrivons un système représentatif, InfoSleuth [BBB+98].

InfoSleuth

InfoSleuth [JB97], [BBB+97] a été créé dans le projet Microelectronics and Computer Technology Corporation (MCC) à Austin, dans le Texas, en 1994. InfoSleuth a pour origine le projet Carnot lui-même développé au MCC, qui vise à constuire des bases de données fédérées afin d'intégrer de façon statique (au sens où les structures de données traitées sont cohérentes et centralisées) des bases d'informations hétérogènes. InfoSleuth se veut plus adapté à un environnement constamment changeant qu'est le Web. InfoSleuth est aujourd'hui commercialisé.

Un agent est autonome et communique avec d'autres agents via le langage KQML au-dessus du protocole HTTP ou TCP/IP. Chaque agent a des capacités particulières. Il interagit avec d'autres agents par exemple en jouant le rôle de fournisseur d'informations, de consommateur d'informations ou les deux. Il peut s'interfacer avec des composants logiciels tels que les BD et les navigateurs. Il peut posséder des capacités d'inférence de la connaissance, par exemple fusionner différentes sources d'infos en une vue cohérente et peut se spécialiser dans des domaines d'expertise.

Un agent utilisateur représente les intérêts des utilisateurs. Il assiste l'utilisateur en formulant des requêtes sur des modèles de domaines communs et affiche les résultats des requêtes selon le contexte de l'utilisateur.

L'agent courtier fournit un service de correspondance sémantique entre l'agent qui cherche un service particulier et l'agent qui fournit ce service. Le courtier supporte la scalabilité et les agents qui fournissent un service s'enregistrent auprès du courtier en spécifiant le langage, et à quels types de questions ils répondent.

L'agent ontologie connaît le réseau de concepts qui décrit chaque domaine d'intérêt et qui fournit le langage commun que parlent agent et utilisateur.

L'agent d'exécution de tâche plannifie et coordonne les requêtes des autres agents. Un plan d'exécution est conçu pour réagir aux événements inattendus.

Un agent de ressource rend l'information contenue dans une BD disponible et à jour. C'est une interface entre un agent et une source de donnée locale. Il cache le langage natif, et parle le langage défini dans l'ontologie commune.

L'agent requête multi-source décompose une requête complexe en sous-requêtes pour déléguer celles-ci aux agents capables d'y répondre et compose les sous-réponses en une réponse à la requête complexe initiale.

L'agent moniteur gère les échanges de messages entre les autres agents, mais ne s'engage pas lui-même.

3.3.8 Whois++

L'architecture de Whois++ [GW95] présente une hiérarchie de serveurs d'index et comprend la notion de centroïde. Le protocole Whois++ a été conçu pour permettre d'interroger une base annuaire et pallier la lourdeur du service d'annuaire existant géré par le protocole X500. Whois++ fournit une architecture pour indexer des bases de données distribuées. Il comprend un modèle de données et un protocole de requête.

Indexation

Le principe d'indexation est le suivant : chaque serveur Whois++ se charge d'indexer des pages selon une certaine structure de données. Les informations se présentent sous la forme d'ensemble d'attribut-valeur. L'attribut étant par exemple type-document, domaine etc... L'intérêt d'utiliser ce modèle est de pouvoir construire une hiérarchie de serveurs d'index, un serveur de haut niveau pouvant indexer des informations sur un serveur de base Whois++ ou bien sur un serveur d'un niveau supérieur. Un tel type de serveur est alors appelé centroïde : son but est de rassembler l'information sur un critère de sélection, en l'occurrence un attribut particulier. Prenons par exemple un serveur qui contient les informations suivantes :

```
Template : user
First Name : Jean
Last Name : Dupont
Town : Saint Etienne

Template : user
First Name : Pierre
Last Name : Dupond
Town : Saint Etienne

Template : user
First Name : Chloé
Last Name : Durand
Town : Paris
```

Le centroïde rassemblant l'information sur le critère de la ville contiendra les données suivantes :

```
Template : user
First Name : Jean Pierre
Last Name : Dupont Dupond
Town : Saint Etienne

Template : user
First Name : Chloé
Last Name : Durand
Town : Paris
```

Interrogation

Chaque requête est envoyée à un serveur d'index dans un format standard, et c'est à la charge du serveur d'index de déterminer la collection de centroïdes susceptibles de répondre à cette requête ou bien de renvoyer au client les adresses d'autres serveurs capables d'orienter la requête. Il y a ainsi propagation de l'information, de proche en proche, jusqu'aux serveurs contenant les documents adéquats. Pour initier la recherche, il existe un serveur spécial contenant le sommaire de l'information et qui permet dans un premier temps de diriger les requêtes vers des serveurs.

Coopération

Les serveurs d'index sont distribués dans le sens où ils indexent de l'information distribuée, répartie sur plusieurs sites distants.

CIP (Common Indexing Protocol) est une extension de Whois++. Les notions de centroïdes et de hiérarchie de serveurs d'index sont conservées, CIP étend la collection de données indexées suivant des template_types particuliers, à des collections de données non formatées.
L'architecture CIP se présente comme suit :
Le processus de recherche d'informations client/serveur inclut les ensembles de données et les protocoles d'accès aux données. Un serveur est responsable des requêtes sur un domaine (ensemble de données). Les clients lancent des recherches sur ces ensembles de données via un protocole d'accès aux données, comme par exemple LDAP ou Whois++, HTTP, Z39.50.
CIP permet de router des requêtes lorsque le serveur auquel le client s'adresse ne connait pas la réponse à une requête, le serveur local cherche d'abord dans sa propre collection de centroïdes. S'il a la réponse, il retourne l'adresse du serveur d'où il a collecté ces centroïdes.

3.4 Synthèse

Cette synthèse compare les systèmes précédents afin d'en dégager les forces et les faiblesses. Cette comparaison est résumée sous la forme de quatre tableaux décrivant respectivement : les caractéristiques générales, le corpus, le gestionnaire de requêtes et le gestionnaire d'index.

Le premier tableau 3.5 présente les caractéristiques générales de chaque système, à savoir l'origine et la date de référence sur laquelle porte l'étude, l'état d'avancement (prototype, produit), le type d'architecture (centralisé, distribué, parallèle) pour l'indexation et la recherche, et les fonctionnalités supplémentaires. Nous constatons que le type d'architecture d'index est centralisé dans la majeure partie des outils de recherche classiques, comme AltaVista, Yahoo !. L'indexation est distribuée et la recherche centralisée, dans le cas de Harvest. Dans notre étude, seul le système multi-agents InfoSleuth apporte un aspect de coopération dans la recherche et dans l'indexation.

Nous nous attachons dans le deuxième tableau récapitulatif (traitement du corpus) à distinguer : la prise en compte de l'hétérogénéité, la structure, la granularité, le volume, et le partage du corpus.

TAB. 3.5 – Caractéristiques générales des SRI

généralités	Altavista	Yahoo !	Meta-crawler	Onto-broker	Harvest	HyPursuit	Pharos	Info-Sleuth
origine	Palo Alto, Californie	université Stanford	université Washington	KarlsRuhe, Allemagne	université Colorado	MIT	université Californie, Santa Barbara	Austin, Texas
référence	1995-2000	1994-2000	1995-	1996	1994-1996	1996	1998	1995-2000
état	produit	produit	produit	prototype	produit	prototype	prototype	produit
indexation	centralisée	centralisée	-	centralisée	distribuée	distribuée	distribuée	distribuée
recherche	centralisée	centralisée	parallèle	centralisée	centralisée	centralisée	distribuée	distribuée
fonctionnalités			fusion de résultats	inférence logique de Frame		clusterisation a priori		

Les outils que nous étudions prennent en entrée un corpus hétérogène, que ce soit le Web avec les outils comme AltaVista, HyPursuit, ou l'Intranet, réservé à InfoSleuth. En effet, les pages collectées forment un amas non structuré, dont la sémantique n'est pas explicite. Cette caractéristique est à l'origine du fort taux de bruit et de silence dans les réponses données par des outils généralistes. Les outils sont comparés selon leur capacité à traiter l'hétérogénéité des données. Les systèmes centralisés, de type AltaVista, n'ont en général pas cette capacité. Par conséquent le problème du bruit et du silence est très important.

Autre élément essentiel des propriétés du corpus, la structure. Cette dernière va conférer au corpus un aspect sémantique primordial qui sera pris en compte dans tout le reste des étapes d'un système de recherche d'informations. Plus un corpus est structuré au départ, plus il est aisé d'en extraire une représentation sémantique et plus l'interrogation gagnera en puissance d'expression. Or le défaut que l'on peut reprocher à la majorité des outils de recherche sur le Web est précisément de ne pas prendre en compte cette caractéristique. HyPursuit a tenté une approche de structuration automatique de site en utilisant une technique d'agrégation des pages qui repose à la fois sur le contenu du texte et sur les liens hypertextes. Malheureusement, les résultats expérimentaux n'ont pas été très significatifs, dans la mesure où les paquets de pages ne donnaient pas des ensembles facilement interprétables, et le manque de méta-informations donnait

lieu à autant d'interprétations diverses.

Pour la plupart des outils sur le Web, la granularité est fixée par la structure physique du document, qui pour le Web est la page HTML. De même, très peu d'outils gèrent le niveau d'indépendance physique et seul HyPursuit gère le niveau d'indépendance logique. C'est la principale cause de l'échec des outils à résoudre les problèmes de bruit et de silence. En effet, l'importance du niveau de représentation a été relevée, car chaque niveau fournit une sémantique, qu'elle soit structurelle ou contextuelle. De même, le manque de niveau de représentation rend difficile l'interprétation du corpus.

La notion de granularité couplée avec la structure d'un document est d'une importance capitale à cause de son impact sur les étapes du processus de RI. En effet, c'est à partir de cette notion que les documents d'un corpus sont liés à un contexte, et ce contexte est le seul moyen d'appréhender un document dans un environnement donné, et de permettre un repérage immédiat du contenu informationnel. Le contexte est aussi un moyen de réduire le taux de bruit et de silence lorsqu'il est pris en compte dans la requête et dans la recherche. La majorité des outils de recherche classiques ne tiennent pas compte de la granularité.

Enfin le partage de corpus est un moyen d'améliorer le taux de couverture du Web, les outils en majorité centralisés sont susceptibles de n'indexer qu'une faible partie du Web, ce qui est une cause de silence.

TAB. 3.6 – Traitement du corpus par les SRI

corpus	Altavista	Yahoo !	Meta-crawler	Ontobroker	Harvest	HyPursuit	Pharos	InfoSleutl
hétérogénéité	non prise en compte	non	non	oui	non	non	non	non
structure	textuelle	complexe	textuelle	complexe	textuelle	hiérarchique	complexe	complexe
granularité	page	page	page	page	fichier	variable	page	page
volume	320M pages	1,8M pages	600M pages	-	-	100 sites		
partage de corpus	non	non	oui	non	oui	oui	oui	oui

Le troisième tableau (gestionnaire d'index) compare les représentants du corpus selon les points suivants : le traitement éventuel effectué sur ce corpus

par le système pour construire les représentants des documents, le format des représentants, les méta-informations, la prise en compte du contexte associé aux documents, la structure de l'index, le volume et l'architecture de l'index.

La construction des représentants s'effectue automatiquement pour tous les outils de recherche, en général par recopie des pages téléchargées au moment de la constitution du corpus à l'aide d'un robot. Notons que pour AltaVista, le représentant est composé du champ de méta-information qui est analysé lorsque la balise META est reconnue dans la page HTML. La majorité des outils n'atteignent pas l'indépendance physique ni l'indépendance logique. Les représentants sont tout au plus des donnés semi-structurées décrites par des méta-informations.

Peu d'outils, hormis les systèmes de bibliothèques utilisent les langages d'indexation contrôlés. L'utilisation du vocabulaire contrôlé est cependant un moyen de réduire le bruit.

De même, les méta-informations ne sont pas utilisées par les outils de recherche. L'usage de la balise META s'est avéré peu pertinent, car alors AltaVista ne prend plus en compte le contenu textuel de la page, et les mots-clés ne donnent qu'un aperçu ponctuel du contenu informationnel d'une page dans un langage libre qui peut prêter à confusion. Excepté HyPursuit et NorthernLight, aucun outil ne prend en compte le contexte. Dans HyPursuit, le contexte est déduit automatiquement par clusterisation à priori, cependant et da façon plus marquée par rapport à l'usage des balises $< META = keywords >$, il manque une sémantique précise et significative aux labels décrivant les clusters, ce sont des mots-clés extraits du langage libre et qui ne sont pas pertinents. Northern-Ligth propose une clusterisation à posteriori, qui repose sur la création d'une hiérarchie de cluster extraite à partir des mots de la requête et dans laquelle sont rangées les réponses trouvées par le moteur de recherche. Là encore, bien que cette solution propose une contextualisation des réponses, il est parfois difficile d'exploiter cette dernière parce qu'elle unifie une idée unique d'un ensemble de pages qui proviennent de n'importe quels sites et prendre en compte cette hétérogénéité est risqué.

En conclusion, la limite actuelle des outils de recherche réside dans un manque de représentativité du corpus qu'ils traitent, et ceci parce que les données du Web manquent de sémantique et de structuration au sein d'un document plus global qu'une page, par exemple un site.

Il y a donc un réel besoin de mieux décrire le contenu et le contexte et d'extraire une structuration des sites Web.

En comparant maintenant les volumes et les types de collections de documents, nous pouvons observer que lorsque le volume d'informations et le type de corpus sont respectivement limités et spécialisés, les outils sont plus complexes et répondent bien aux besoins utilisateurs en termes de performance et d'efficience. Au contraire, dans le cas d'un corpus hétérogène et volumineux, les outils simples, génériques et efficaces sont utilisés mais ne donnent pas satisfaction aux utilisateurs. À ce jour, aucun outil sur le Web ne propose une architecture qui permettrait d'intégrer les outils spécialisés, performant et efficients ciblés sur des portions de données disponibles sur le Web.

La majorité des outils utilisent les index en fichiers inversés, stockés dans des bases de données ou systèmes de fichiers. Cela s'explique par des raisons de facilité de mise en œuvre de l'index et d'efficacité dans la phase de correspondance.

La centralisation de l'indexation provoque le silence, induit par le faible taux de couverture, et le bruit, induit par des délais de raffraichissement d'index longs.

TAB. 3.7 – Gestionnaire d'index

représentants	Altavista	Yahoo !	Metacrawler	Ontobroker	Harvest	HyPursuit	Pharos	InfoSleutl
construction	automatique	manuelle	-	manuelle	automatique	automatique	manuelle	manuelle
format	HTML	HTML	-	logique frame	SOIF	textuel	MARC	
méta-informations	$< META >$	oui	non	oui	oui	non	oui	oui
contexte	non	non	non	non	non	oui	non	non
structure de l'index	fichier inverse	fichier inverse	-	base données	fichier inverse	fichier inverse		BD
volume de l'index	100%	5%		$< 100\%$	jusque 200%	100%		
architecture	centralisée	centralisée	distribuée	centralisée	distribuée	distribuée	distribuée	distribuée

Le quatrième tableau synthétise les caractéristiques de l'interface utilisateur et du gestionnaire de requêtes. L'interface utilisateur est caractérisée par le langage de requêtes qui peut être plus ou moins structuré, l'utilisation des méta-informations dans la formulation des requêtes ainsi que la présentation

des résultats et le type d'architecture. Le gestionnaire de requêtes gère la fonction de correspondance entre la requête et la base d'index ainsi que l'architecture de la recherche.

En général, les outils de recherche proposent une interface d'interrogation par mots-clés associés à des opérateurs booléens et un opérateur d'exclusion ainsi que l'usage des parenthèses. Certains outils comme Altavista ou Harvest offrent la possibilité de parenthéser les expressions booléennes, d'interroger des expressions exactes, d'utiliser des troncatures ou des jockers. Inquery offre une interface plus évoluée en langage naturel. Harvest, propose une possibilité d'interroger les balises indexées, ce qui donne un pouvoir d'expression supplémentaire à la requête.

Presque tous les systèmes de recherche proposent une présentation des résultats sous la forme d'une liste de références aux documents retrouvés. Cependant, des outils comme NorthernLight clusterisent les réponses obtenues en fonction des termes de la requête et fournissent donc une catégorisation des réponses. Excite utilise une analyse sémantique pour regrouper des réponses selon des thèmes. C'est le cas également de HyPursuit qui clusterise les documents a priori et se sert de cette clusterisation pour afficher des réponses hiérarchisées. Cependant les hiérarchies de concepts induites par les hiérarchies de clusters sont difficilement interprétables.

La fonction de correspondance utilise très souvent le modèle booléen, qui s'adapte tout à fait à la structure d'index en fichier inverse. Notons que Google prend en compte dans sa fonction de correspondance le nombre de liens pointant sur les pages réponses pour établir un classement.

Les systèmes qui introduisent le routage dans leur architecture de recherche répondent au problème de scalabilité.

TAB. 3.8 – Gestionnaire de requêtes

langage de requête	Altavista	Yahoo !	Meta-crawler	Ontobroker	Harvest	HyPursuit	Pharos	InfoSleutl
méta-informations	balises HTML	champs structurés	non	attributs	balises HTML	ontologie	ontologie	
présentation des résultats	liste triée	liste hiérarchisée	liste triée	liste d'objets	liste triée	clusters	liste	
architecture	centralisée redondante	centralisée	parallèle	centralisée	distribuée	distribuée avec routage	distribuée avec routage	distribuée

Chapitre 4

Vers une structuration des ressources sur l'Internet : les documents Web

Dans les précédents chapitres, nous avons identifié le problème du bruit et du silence dans les outils de recherche d'informations. Dans le cadre de ce chapitre, nous proposons une approche basée sur la structuration de documents pour tenter de remédier à ce problème.

Tout d'abord nous introduisons la notion de documents et l'intérêt des documents structurés. Puis nous définissons le concept de *document Web* ainsi que l'organisation des documents Web en les illustrant à travers deux exemples, l'un est celui du document thèse, l'autre celui du site AGORA21, serveur spécialisé dans le développement durable, hébergé à l'école des Mines de Saint-Etienne. Nous formalisons la représentation et la description de ces documents. Enfin, nous expliquons comment les documents Web s'intègrent dans un système de recherche d'informations, plus précisément dans les phases de création, d'indexation et de recherche de documents.

4.1 Qu'est-ce qu'un document ?

La définition du Petit Robert pour le mot document est ¡¡ Tout écrit servant de preuve ou de renseignement ¿¿.

Dans le monde de l'imprimerie, un document est facilement assimilé à sa structure physique, par exemple un livre, un article, un journal. De plus, un document possède généralement sa propre structure logique, par exemple un livre peut être découpé en chapitres, sous-chapitres, puis en sections et paragraphes. La table des matières ainsi que le résumé d'un document permettent déjà d'appréhender le contenu sémantique ainsi que le contexte et l'organisation des différentes parties du document.

Dans le monde du numérique, un document peut être vu comme un élément d'information stocké sur un support physique. Sa structure physique est délimitée par sa représentation physique, par exemple une page HTML, un fichier informatique.

Cependant, un document représente un ensemble cohérent d'informations traitant en général d'un même thème ou regroupées selon certains critères. Par exemple une revue informatique, le journal *Le Monde* sur le thème des actualités, un annuaire contenant les noms et adresses par ordre alphabétique des individus d'un même département, un recueil de poèmes.

Une observation importante concerne la granularité de l'information. En effet, une unité d'information utile peut être réduite à un paragraphe, ou être étendue à un ensemble de fichiers, une collection de pages. La manipulation et l'accès à ces unités d'informations deviennent plus complexes.

La notion de document recouvre donc selon les usages et les besoins des caractéristiques d'unité, de structuration, de support. Pour ce qui nous concerne, nous définissons de manière informelle un document comme tout ce qui peut être considéré comme pertinent par rapport à un besoin d'information d'un utilisateur, et nous laisserons le soin à l'auteur ou l'éditeur des documents de choisir et d'expliciter ces unités d'informations.

Notre choix est en fait une sorte d'explicitation des processus de rédaction et de publication (édition). En effet, un auteur fait toujours des choix à plusieurs niveaux : le choix d'écrire tel livre, sur tel sujet, pour tel public, etc. Il fait ensuite des choix de structures : découpage en chapitres, sections, paragraphes. Au niveau de la publication, l'éditeur fait le choix de créer un journal sur telle ou telle thématique, de regrouper tel ou tel articles, de publier tel livre, de proposer une nouvelle édition. À chaque fois, un document dans le sens où nous l'entendons est pensé par l'auteur ou l'éditeur.

L'intérêt de manipuler des documents tels que nous venons de les définir est de pouvoir traiter l'information au degré de détail adéquat et de fournir des réponses plus pertinentes aux questions.

4.2 Démarche

Dans notre état de l'art, nous avons cité deux catégories d'outils de recherche sur le Web. La première catégorie regroupe les outils de recherche dits en texte intégral. Rappelons que les avantages de ces outils sont la facilité de la mise en œuvre des index et la rapidité de la recherche dans la base d'index. L'importance de ces qualités font que les outils de recherche en texte intégral sont encore les plus utilisés pour retrouver de l'information sur le Web. La deuxième catégorie d'outil, les annuaires thématiques reposent sur une classification a priori des sites selon un schéma de classification ad-hoc établi manuellement par des experts et des linguistes. Cependant, les limites évoquées dans létat de l'art montrent que ces outils sont encore loin d'avoir résolu les problèmes spécifiques au Web. Nous présentons les apports de notre approche face à ces limites.

4.2.1 Spécification du contenu sémantique

La recherche d'information se tourne principalement vers une automatisation de la recherche en texte intégral, qui préconise la manipulation des seuls termes présents dans le corpus, ce afin de préserver la richesse de la langue naturelle. Nous pouvons citer des outils de classification automatique [DAAP98], [CGRU97]. Cependant cette démarche résout modestement l'ambiguité des mots et des contextes.

Dans notre travail, nous optons pour une double représentation du contenu, qui consiste à prendre en compte la totalité du contenu textuel du document et à utiliser la description de ce contenu faite par son auteur à l'aide de mots-clés univoques. Pour assurer cette univocité des termes dans les phases de représentation, d'indexation et d'interrogation, nous utiliserons des modèles de classification et de thésaurus. Notons que l'utilisation d'outils de traitement de la langue naturelle et de classification automatique tels ceux que nous avons cités ne sont exclus ni dans notre modèle de documents Web ni dans notre architecture.

4.2.2 Spécification de la structure

De récents travaux dans le domaine de la recherche d'information ont montré que la prise en compte de la structure dans les différentes étapes du processus de recherche d'information apportait une amélioration notable dans la pertinence des réponses d'un système de recherche d'information [AD99], [ADB99a], [KKA].

Nous avions déjà noté que la structure d'un document conférait une certaine sémantique au document, et qu'elle aidait le lecteur à appréhender l'information contenue dans ce dernier. Parmi tous les outils de recherche d'information sur le Web, que ce soient des annuaires ou des outils de recherche en texte intégral, aucun ne prend en compte la structure globale d'un ensemble de pages Web. Notre apport consiste à expliciter la structure de l'information contenue dans les sites Web, et de montrer que hiérarchiser les documents permet à la fois de réduire le bruit et le silence mais aussi d'accéder plus rapidement à l'information pertinente sur le Web.

Lié à la structure, le contexte que nous décrivons dans le paragraphe suivant est également un point important que nous traitons dans notre modèle de documents Web.

4.2.3 Spécification du contexte

Nous considérons que toute information pour être appréhendée efficacement doit se trouver dans un contexte, c'est-à-dire connectée à des informations d'ordre plus général qui la précisent et permettent de situer le cadre dans lequel elle se situe.

Par exemple lorsque nous parlons du traitement de l'eau nous pouvons situer le contexte de la pollution et de l'environnement. Ce contexte aurait pu être la chimie ou les usines de dépollution de l'eau. Notons que le contexte, tel que nous le définissons en section 2 de ce chapitre, désigne non seulement le domaine de connaissance, mais également toute autre information significative, comme la date, l'organisation, le type de donnée. La spécification du contexte dans notre modèle de documents Web apporte les points suivants :

- un contexte est explicité par l'auteur des pages ou par un expert,
- lors de la consultation d'informations, un utilisateur visualise les hiérarchies de contextes,
- enfin lorsqu'une réponse est fournie par un système de recherche d'informations, elle doit se présenter sous la forme d'une information (au sens document) précédée de son contexte, c'est-à-dire avec la structure logique de l'information ou la ou les arborescences dont elle constitue un nœud.

4.2.4 Prise en compte de l'hétérogénéité

A l'heure actuelle, il n'existe pas d'outils capables de généraliser l'indexation automatique des données multimédia sur le Web. Nous prenons en compte cette

hétérogénéité par l'ajout des méta-informations pour décrire aussi bien les pages Web que les ressources de tout type disponibles sur l'Internet. Les autres types d'étérogénéité que nous avons vu dans le chapitre 2 seront également traités grâce aux méta-informations. Dans la suite de ce chapitre, nous détaillerons la mise en œuvre de cette solution.

4.2.5 Choix de la fonction de correspondance

Dans le chapitre bibliographie, nous avons détaillé les différentes fonctions de correspondance utilisées par les systèmes de recherche d'information, et notamment le modèle booléen privilégié par les outils de recherche sur le Web. En effet, les modèles vectoriels et statistiques donnent les meilleures performances en terme d'efficience (meilleurs taux de rappel et de précision). Cependant pour des raisons d'efficacité, au vu du volume d'informations traitées et du nombre moyen de termes utilisées dans une requête (2,3 mots selon les dernières statistiques []), le modèle booléen répond le mieux aux exigences du Web.

Dans nos expérimentations du modèle de documents Web, nous choisissons le modèle booléen pour des raisons de simplicité de mise en œuvre du prototype. Nous manipulerons des données semi-structurées, c'est-à-dire des méta-informations qui décrivent les documents Web ainsi que des données non structurées, les données textuelles des pages Web. Nous expliquerons ensuite l'algorithme qui nous sert à calculer si un document est pertinent par rapport à une question.

4.2.6 Spécification des vues sur les documents

Une vue correspond dans notre définition à une structuration d'un ensemble de pages, ce qui corrobore aussi la définition de document Web. Cette définition part de la constatation suivante : Lorsqu'un auteur crée un site, il a en général une vision hiérarchique de son site, que l'on pourrait appeler vue. Même si de milliers de liens mettent en relation les pages de ce site, elles gardent une organisation bien précise, prédéfinie, et l'auteur sait où retrouver ses informations.

Or un utilisateur peut, en regardant un site, sélectionner ou filtrer uniquement l'information qui l'intéresse, auquel cas il peut réorganiser l'information selon son point de vue. On peut dire qu'il a une autre vue d'un même ensemble de pages. Ces vues sont virtuelles et non explicitables. Il serait en effet irréalisable de créer à-priori autant d'organisations de documents qu'il existe de vues des utilisateurs et des auteurs de sites. Cependant, certains outils essaient

de classifier à-postériori les pages résultant d'une interrogation en des catégories qui correspondraient au profil d'un utilisateur, voir les efforts accomplis par Northern Light, WebMate. La classification se limite à un niveau et est en général très approximative.

Nous choisissons de nous limiter à la vue d'un auteur en lui permettant d'expliciter son organisation de l'information. Ainsi, une des vues accessible à l'utilisateur en réponse à sa question sera celle de l'auteur. Nous pouvons rendre accessibles d'autres vues, par exemple si nous utilisons la vue d'un spécialiste qui organisera les documents Web selon une classification de concepts générale ou spécialisée, ou bien si nous utilisons un outil de classification automatique de pages Web pour construire les documents Web.

4.2.7 Utilisation de classification normalisée et de thésaurus

La caractéristique essentielle des annuaires est de fournir une classification de termes sous la forme d'une arborescence de nœuds, où chaque nœud peut en réalité référencer un autre nœud, il s'agit d'un DAG. Cette classification est universelle, dans le sens où l'ensemble des concepts est censé traduire toute la connaissance, mais non standarde; chaque annuaire a sa propre classification de termes généralement établie par des linguistes, comme c'est le cas pour Yahoo ! Chaque annuaire a donc sa propre vision d'une organisation de la connaissance, et fournit une classification des sites Web au sein de cette arborescence de termes. Il est à noter qu'un auteur de site peut choisir l'endroit où son site apparaîtra dans la classification en envoyant un formulaire dans lequel il fournit un résumé et une description de son site. Du côté utilisateur, l'interrogation portera alors uniquement sur le résumé et la desciption fournie par l'auteur.

Nous pouvons faire l'analogie avec les méta-informations inclues dans les pages Web par son auteur avec la balise META et qui seront les seules informations indexées et donc interrogeables pour certains outils de recherche, par exemple Alta Vista.

Dans les deux cas, il s'agit de réduire le bruit et d'accéder plus rapidement aux informations pertinentes. Yahoo ! est utilisé pour obtenir des informations générales sur des thèmes précis. A la différence de Yahoo !, nous indexons les pages Web en texte intégral, ce qui permet de garder une interrogation le contenu des pages. L'utilisation du thésaurus GEMET1.5 dédié au développement durable nous permet d'indexer les documents Web en utilisant un vocabulaire d'indexation normalisé, ce afin de réduire le bruit et le silence. Le thésaurus est

accessible via l'interface de requête pour aider l'utilisateur à formuler sa requête. Nous offrons une indexation plus fine que celle se limitant à des pages choisies d'un site, tout en permettant une interrogation combinée sur les descripteurs et le texte.

4.2.8 Résumé

Pour résumer les points ou limites pour lesquels nous apportons des solutions grâce à notre modèle de documents Web, nous retiendrons :

1 indexation coûteuse lorsque la granularité est fixe, celle de la page HTML (intégralité de la page téléchargée et indexée)
2 pas de contrôle sur le vocabulaire d'indexation
3 le contexte (domaine ou sujet dans lequel se trouve une page) n'est pas explicite ni utilisé
4 pas de notion de structure logique de document (titre, paragraphe, résumé, parties de documents, etc.)
5 L'Internet contient par essence des documents de tous types, c'est-à-dire hétérogènes par leur format (video, son, image, texte) et leur domaine d'application (données quantitatives, qualitatives, logistiques, normatives, etc.). Ces caractéristiques ne sont pas traitées par les outils de recherche actuels.

Nous définissons à présent la notion de document et plus particulièrement de document Web.

4.3 Les documents Web : approche et illustration

Un document Web est créé par un auteur parce qu'il correspond à un besoin d'expression de celui-ci, sa structure logique est déteminée par son auteur. Un document Web représente une ressource Web, une page Web, un site Web, ou un système de recherche sur le Web. Il est caractérisé par un identifiant, des liens structurels, des méta-informations et un contexte où :

- L'identifiant est sa localisation sur un site définie par l'adresse internet (URL). Cet identifiant doit être unique dans toute l'organisation des documents Web.
- Les liens structurels définissent l'organisation sous forme de DAG (Direc-

ted Acyclic Graph) de l'ensemble des documents Web, [DAB99]. Il existe 4 types de liens structurels : les liens de composition les liens de référence, les liens de séquence, et les liens de contenu.

- *Les liens de compositions* sont les liens hiérarchiques entre les documents Web. Le lien *fils* définit la relation entre un document père et les documents qui le composent ; le lien *père* est la relation inverse. Par exemple le document thèse se compose des 6 documents introduction, état de l'art ... grâce au lien fils.
- *Les liens de séquence* définissent le séquencement ou la lecture linéaire de deux documents Web. Le document d21 et d22 sont reliés par un lien de séquence.
- *Les liens de référence* définissent une relation de référençage entre deux documents Web. Le document biblio est relié au document d2 par un lien de référence.
- *Les liens de contenu* définissent l'appartenance des pages à un document Web. Les pages p1 à p16 appartiennent au document d1.

- Les méta-informations décrivent la sémantique d'un document Web sous la forme d'un ensemble de champs structurés appelés descripteurs.
- Le contexte qui permet à tout document d'hériter des méta-informations de tous les ancêtres du nœud de l'arbre dans lequel il se trouve.

Pour illustration, prenons deux exemples. Le document thèse, exemple 1, figure fig.4.1 est la fabrication d'un document hypertexte à partir d'un document structuré en latex, possédant toutes les caractéristiques propres à un livre : titre, auteur, identifiant, découpage en chapitres et parties logiques, bibliographie. Le document thèse est supposé accessible à l'adresse `http://groseille.emse.fr:8000/THESE`.

- Par exemple, pour obtenir le document thèse, il suffit d'accéder à l'adresse `http://groseille.emse.fr:8000/THESE/d0.xml`, les sous-documents sont aux adresses relatives d1.xml, d2.xml ...
- Les documents Web sont organisés en DAG. L'organisation des documents Web s'effectue par une opération d'extraction de la structure logique, dans la phase de traitement du corpus. Par exemple, dans la figure fig.4.1, la table des matières induit une organisation hiérarchique de la thèse. Cette organisation correspond au niveau de représentation logique.

  Cette organisation semble naturelle, vu la définition hiérarchique de la thèse, cependant, lorsque les critères de méta-informations seront définis, des opérateurs permettront d'extraire un niveau de représentation externe

à partir de cette structure logique de base.

- Les méta-informations sont écrites en XML, les champs admettent des qualificateurs dans le but de les préciser, comme dans Dublin Core :

```
<?XML version=''1.0'' encoding=''iso-8859-1'' standalone=''no''?>
<identifiant>#doc1</identifiant>
<title>La recherche d'informations sur le Web</title>
<author scheme='e-mail' role='auteur principal'>doan@emse.fr</author>
```

- Le contexte de la page 1 de la thèse est représenté par les méta-informations associées à l'introduction de la thèse. Le contexte des pages p18 à p36 rassemble l'union des méta-informations des documents définition et état de l'art de la thèse.

Le deuxième document, exemple 2 en figure fig.4.2 représente le site `http://www.Agora21.org` dont une première organisation est extraite à partir du plan de site faisant apparaître deux niveaux de hiérarchies.

```
Bibliothèque
        Documentation pédagogique
        Publications d'Agora21
        Editions
        Encyclopédie du développement durable
Actualité
    ----
Acteurs
Environnement                                                    biodiv
        climat
        désertification
        eau
        forêts
```

Exemple 2

Liens sémantiques

Les liens sémantiques définissent la relation entre un document et un élément descriptif. Cet élément appartient soit à une méta-information, soit à une ontologie. Un type de lien sémantique correspond à chaque attribut de méta-inforamtion, par exemple le type est-créé-par relie un document à son auteur. Un document est relié à l'ontologie GEMET1.5 par le lien est-décrit-par. Exemple

de méta-information en XML à partir du document thèse :

```
<?XML version=''1.0'' encoding=''iso-8859-1'' standalone=''no''?>
<identifiant>#doc1</identifiant>
<title>La recherche d'informations sur le Web</title>
<author>Bich-Liên Doan</author>
```

Contexte

Le contexte est défini à partir d'une combinaison des liens structurels et des liens sémantiques. Le contexte de d1 du document thèse est l'ensemble de tous les ancêtres à partir du document qui est lié à d1 par la relation de structure. Ici, ce sont les méta-informations du document racine d0.

4.3.1 Vue générale du modèle de documents Web

Nous considérons un seul type d'objet qui est le document Web. Un document Web est vu comme des ensembles de pages Web reliés entre eux par des liens sémantiques (hiérarchiques, séquentiels).

Un document Web a une propriété importante, son niveau de granularité. Il est défini par le niveau dans lequel se trouve le document au sein de la hiérarchie de documents. Plus un document est situé bas dans la hiérarchie, plus sa granularité est fine.

Sous l'aspect de cette modélisation, un site Web peut être considéré comme un document Web, dont la granularité est la plus grosse.

Le schéma fig. 4.3 représente les trois niveaux de notre modélisation du Web.

Le premier niveau représente les pages Web reliées entre elles par des liens hypertextes. Les pages Web ont une structure physique, correspondant à un découpage en pages HTML. Le format HTML permet de décrire la structure logique d'une même page, grâce aux balises représentant les sections et sous-sections de la page ($< H1 >, < H2 > \ldots$), la balise paragraphe $< P >$. Cependant, HTML ne permet pas de représenter la structure logique d'un ensemble de pages, en ce sens que lorsqu'une page est une sous-partie d'une autre page, il n'y a aucun moyen de l'expliciter.

Le deuxième niveau représente notre modèle de documents Web. Un document Web, au contraire d'une page HTML peut modéliser une unité logique d'information. Un document Web est composé d'un ensemble d'unités physiques, les pages Web et d'un ensemble d'unités logiques, elles-mêmes des documents

Web. Nous verrons par la suite que la structure logique d'un document Web sera explicitée grâce au langage XML. La transition entre le premier et le deuxième niveau se fait par regroupements de pages en documents Web et par création de liens typés (structurels, référentiels ou autres) entre les différents documents Web pour en faire des DAGs.

Le troisième niveau représente les hiérarchies de concepts issues des différentes classification de concepts ou thésaurus, et qui servent à décrire les contextes des documents Web, dans un domaine de connaissance précis. Il est à noter que les hiérarchies de documents Web induisent à leur tour des hiérarchies de concepts qui sont différentes de celles issues d'une standardisation.

4.3.2 Exemple

Afin d'illustrer notre propos, nous choisissons comme exemple de document structuré la présente thèse, disponible à la fois sous le format Latex et sous le format HTML. Précisons qu'elle a tout d'abord été écrite sous le format LaTeXpuis transformée en HTML via le programme Latex2Html.

Nous schématisons ci-dessous la structure de cette thèse, aisément extraite puisque la structure d'une thèse, comme celle d'un livre est en grande partie linéaire et hiérarchique. Nous analysons la table des matières afin d'en extraire une première décomposition structurelle. Nous en tirons les documents Web et les liens de structure, et nous faisons apparaître d'autres liens typés qui n'étaient pas explicités dans l'organisation initiale du document thèse.

La partie haute de la figure fig.4.1 représente les différents chapitres de la thèse ainsi que les liens entre ces chapitres qui sont des liens de séquence, de référence et d'association sémantique. Cette structure est en partie reprise dans la partie droite de la figure, qui représente une partie de la hiérarchie de documents Web, où figurent en gros traits les liens de structure. Pour décrire les documents Web, nous avons choisi le langage XML, d'une part parce qu'il s'adapte bien à l'écriture de données structurées et de la sémantique associée à ces données, ainsi quà la flexibilité de notre modèle de documents Web. D'autre part parce qu'il constitue un nouveau standard dans l'échange des données sur le Web et qu'il fournit par là un langage de communication adéquat dans le cadre de notre architecture coopérante.

4.4 Modélisation des documents Web

Dans cette section, nous proposons une approche formelle pour représenter le World Wide Web ainsi que les *documents Web* , [DB99]. Nous introduisons un ensemble de définitions suivies d'un exemple et d'une représentation en XML.

4.4.1 Modèle du Web

Une *ressource Web* est identifiée et accessible à travers son URL (*Uniform Resource Locator*, cf. RFC 1738 "URL"). Le contenu d'une ressource Web peut être du texte ASCII, du texte respectant le format HTML, les pixels d'une image, du "binaire", etc., et l'élément `Content-Type` permet dans le protocole HTTP de spécifier ce contenu.

Definition 1 *Une ressource Web est un quintuplet $r = (u, c, t, s, d)$, où :*
- *u est l'URL de la ressource définie par le RFC 1738 ;*
- *c est le contenu de la ressource (cf. RFC 2616 "HTTP") ;*
- *t est le format MIME de la ressource (cf. RFC 1341 "MIME") ;*
- *s et d sont respectivement la taille et la date de dernière modification de la ressource (cf. RFC 2616).*

Notations :

On notera :
- $\mathcal{R}$ l'ensemble des ressources du Web ;
- $\mathcal{R}_t$ l'ensemble des ressources du Web dont le type est t ;
- $\mathcal{U}$ l'ensemble des URL des ressources du Web ;
- *www* l'application de $\mathcal{U}$ dans $\mathcal{R}$ qui associe une ressource Web à son URL.

Le cas particulier des *pages Web* est de toute première importance, puisque c'est grâce à elles que le Web est un hypertexte.

Definition 2 *Une page Web est une ressource Web dont le type est* `text/HTML`*, c'est-à-dire un élément de $\mathcal{R}_{\texttt{text/HTML}}$.*

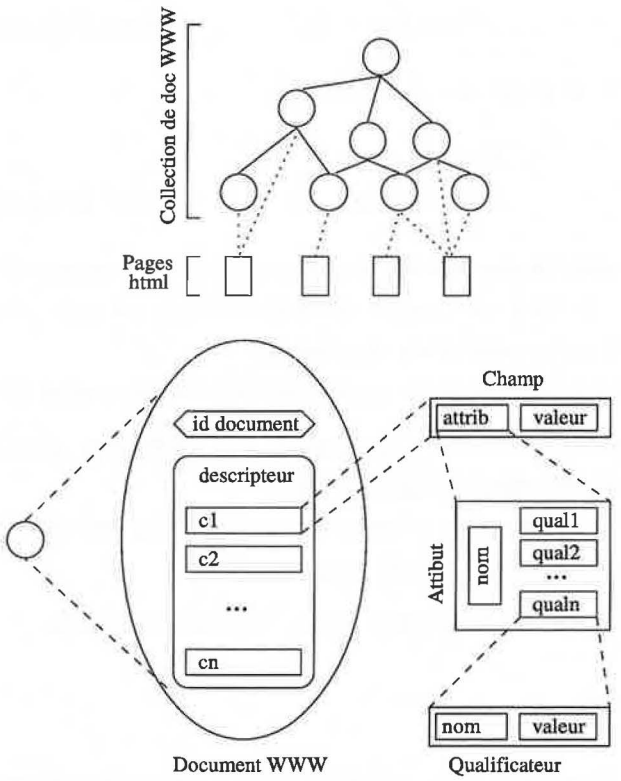
Definition 3 *Un lien hypertexte est un triplet $lh = (source, label, target)$, où :*
- *source est l'URL de la ressource d'où le lien part ;*
- *label est le texte attaché au lien (l'ancre) ;*
- *et target est l'URL de la ressource de destination.*

4.4.2 Modèle des documents Web

Dans le cadre de ce chapitre, nous présentons formellement la notion de documents Web. Dans un premier temps nous présentons succinctement les éléments du modèle, puis nous définirons précisément chacun de ces éléments.

Un document d est constitué d'un ensemble de *champs* que l'on appellera *descripteur* par la suite, noté δ_d ainsi que d'un identificateur noté id_d. Chaque champ c est composé d'un *attribut* a_c et d'une *valeur* v_c. Un attribut a est identifié par un *nom* n_a et un ensemble de *qualificateurs* Q_a. Un qualificateur est un couple *nom* n_q et *valeur* v_q.

Les documents Web sont organisés en un graphe orienté acyclique, nous formaliserons les relations structurelles entre documents. Un document d est associé à un ensemble de ressources Web, mais pour faciliter la mise en œuvre de notre application, nous considérons qu'il s'agit d'un ensemble de pages Web inclus dans $\mathcal{R}_{text/HTML}$.



Soit $\mathcal{N}$ un ensemble de noms.
Soit $\mathcal{V}$ un ensemble de valeurs.

On peut par exemple prendre pour $\mathcal{N}$ et $\mathcal{V}$ l'ensemble des chaînes de caractères, noté $\mathcal{L}_{String(8859-1)}$.

L'ensemble des noms est utilisé pour les identifiants de documents, les noms de champs (ou noms d'attributs) et les noms de qualificateurs. L'ensemble des valeurs est utilisé pour les valeurs de champs et les valeurs de qualificateurs.

Definition 4 *Un document Web est un couple* $d = (i_d, \delta_d) \in \mathcal{N} \times \Delta$ *où :*
- $i_d \in \mathcal{N}$ *est l'identifiant d'un document,*
- *et* $\delta_d \in \Delta$ *est son descripteur.*

Notations :

On appelle :
- $\mathcal{D}$ l'ensemble $\mathcal{N} \times \Delta$ des documents Web,
- id la projection de $\mathcal{D}$ dans $\mathcal{N}$,

$$id : \mathcal{D} \longrightarrow \mathcal{N}$$
$$id((i_d, \delta_d)) = i_d, \qquad \text{pour tout } (i_d, \delta_d) \in \mathcal{D}$$

- $desc$ la projection de $\mathcal{D}$ dans Δ,

$$desc : \mathcal{D} \longrightarrow \Delta$$
$$desc((i_d, \delta_d)) = \delta_d, \qquad \text{pour tout } (i_d, \delta_d) \in \mathcal{D}$$

Le corpus de documents est déterminé par les ressources Web (pages HTML, images ...) ainsi que par les documents Web qui sont identifiés de manière unique et qui possèdent un descripteur.

Voici un exemple de document décrit en XML, extrait du document thèse.

```
<?XML version=''1.0'' encoding=''iso-8859-1'' standalone=''no''?>
<identifiant>#doc1</identifiant>
<title>La recherche d'informations sur le Web</title>
<author>Bich-Liên Doan</author>
```

Definition 5 *Un descripteur est un ensemble de champs.*

Notations :

On appelle :
- Δ l'ensemble $\mathcal{P}(\mathcal{C}) = \mathcal{P}(\mathcal{N} \times \mathcal{P}(\mathcal{Q}) \times \mathcal{V})$ des descripteurs.
- Pour tout attribut a, nous définissons la fonction $val_a : \Delta \longrightarrow \mathcal{P}(\mathcal{V})$, qui à tout descripteur $\delta \in \Delta$ associe l'ensemble des valeurs $v \in \mathcal{V}$, telles que

$(a, v) \in \delta$:

$$val_a : \Delta \longrightarrow \mathcal{P}(\mathcal{V})$$
$$val_a(\delta) = \{v \in \mathcal{V} \mid (a, v) \in \delta\}$$

- De même, pour tout attribut $a = (n_a, Q_a) \in \mathcal{A}$, de qualificateurs $Q_a \subset \mathcal{Q}$, nous définissons la fonction $\overline{val_a} : \Delta \longrightarrow \mathcal{P}(\mathcal{V})$, qui à tout descripteur $\delta \in \Delta$ associe l'ensemble des valeurs $v \in \mathcal{V}$, telles que $((\exists Q \supset Q_a)((a, Q, v) \in \delta))$:

$$\overline{val_a} : \Delta \longrightarrow \mathcal{P}(\mathcal{V})$$
$$\overline{val_a}(\delta) = \bigcup_{Q \supset qual(a)} val_{(nom(a),Q)}(\delta)$$

Les fonctions val_a retournent l'ensemble des valeurs des champs qui ont exactement pour attribut a, et les fonctions $\overline{val_a}$ retournent l'ensemble des valeurs des champs qui sont surqualifiés par rapport à l'attribut a.

Definition 6 *Un qualificateur est un couple $q = (n_q, v_q) \in \mathcal{N} \times \mathcal{V}$ où :*

- *n_q est le nom du qualificateur,*
- *et v_q sa valeur.*

Conformément à Dublin Core, nous introduisons ici la notion de qualificateur dans le but de préciser un champ.

Exemples :

- q_0 = ("scheme", "e-mail")
- q_1 = ("role", "premier auteur")

Notations :

On appelle :

- $\mathcal{Q}$ l'ensemble $\mathcal{N} \times \mathcal{V}$ des qualificateurs,
- *nom* la projection de $\mathcal{Q}$ dans $\mathcal{N}$

$$nom : \mathcal{Q} \longrightarrow \mathcal{N},$$
$$nom((n_q, v_q)) = n_q, \qquad \text{pour tout } (n_q, v_q) \in \mathcal{Q}$$

- *valeur* la projection de $\mathcal{Q}$ dans $\mathcal{V}$

$$valeur : \mathcal{Q} \longrightarrow \mathcal{V},$$
$$valeur((n_q, v_q)) = v_q, \qquad \text{pour tout } (n_q, v_q) \in \mathcal{Q}$$

Ces deux fonctions renvoient respectivement les nom et valeur d'un qualificateur q.

Definition 7 *Un attribut est un couple* $a = (n_a, Q_a) \in \mathcal{N} \times \mathcal{P}(\mathcal{Q})$ *où :*

- n_a *est le nom de l'attribut* a,
- *et* Q_a *est l'ensemble des qualificateurs de l'attribut* a, $Q_a \subset \mathcal{Q}$.

Exemples :

- a_0 = ("creator", ϕ)
- a_1 = ("creator", {("scheme", "e-mail"), ("role", "premier auteur") })

Notations :

On appelle :

- $\mathcal{A}$ l'ensemble $\mathcal{N} \times \mathcal{P}(\mathcal{Q})$ des attributs,
- *nom* la projection de $\mathcal{A}$ dans $\mathcal{N}$,

$$nom : \mathcal{A} \longrightarrow \mathcal{N}$$
$$nom((n_a, Q_a)) = n_a, \qquad \text{pour tout } (n_a, Q_a) \in \mathcal{A}$$

- *qual* la projection de $\mathcal{A}$ dans $\mathcal{P}(\mathcal{Q})$,

$$qual : \mathcal{A} \longrightarrow \mathcal{P}(\mathcal{Q})$$
$$qual((n_a, Q_a)) = Q_a, \qquad \text{pour tout } (n_a, Q_a) \in \mathcal{A}$$

Ces deux fonctions renvoient respectivement le nom et l'ensemble des qualificateurs d'un attribut.

Definition 8 *Un champ est un couple* $c = (a, v) \in \mathcal{A} \times \mathcal{V}$ *où :*

- $a \in \mathcal{A}$ *est l'attribut du champ* c,
- *et* $v \in \mathcal{N}$ *est la valeur du champ* c.

Exemples :

- c_0 = (("creator", ϕ), "doan")
- c_1 = (("creator", {("scheme", "e-mail")}), "doan@emse.fr")

Notations :

On appelle :

- $\mathcal{C}$ l'ensemble $\mathcal{A} \times \mathcal{V} = \mathcal{N} \times \mathcal{P}(Q) \times \mathcal{V}$ des champs.

- *attr* la projection de $\mathcal{C}$ dans $\mathcal{A}$

$$attr : \mathcal{C} \longrightarrow \mathcal{A}$$
$$attr((a,v)) = a, \qquad \text{pour tout } (a,v) \in \mathcal{C}$$

- *val* la projection de $\mathcal{C}$ dans $\mathcal{V}$

$$val : \mathcal{C} \longrightarrow \mathcal{V}$$
$$val((a,v)) = v, \qquad \text{pour tout } (a,v) \in \mathcal{C}$$

Par abus de langage, on désignera par *nom de champ* le *nom de l'attribut de ce champ.*

Le document Web thèse peut s'écrire :

```
<?XML version=''1.0'' encoding=''iso-8859-1'' standalone=''no''?>
<identifiant>#doc1</identifiant>
<title>La recherche d'informations sur le Web</title>
<author scheme='e-mail'>doan@emse.fr</author>
```

Le document Web thèse peut être complété :

```
<?XML version=''1.0'' encoding=''iso-8859-1'' standalone=''no''?>
<identifiant>#doc1</identifiant>
<title>La recherche d'informations sur le Web</title>
<author scheme='e-mail' role='auteur principal'>doan@emse.fr</author>
```

Dans notre modèle de document Web nous avons besoin de définir des domaines de valeurs, soit pour les valeurs possibles pour un champ donné, soit la liste des noms de champs permis, ou encore la liste des noms de qualificateurs acceptables pour un nom d'attribut donné. Pour cela, nous introduisons la notion de *langage.*

Definition 9 *Un langage $\mathcal{L}$ est un sous-ensemble de $\mathcal{V}$.*

Exemples :

Nous donnons quelques exemples de différents langages :

- $\mathcal{L}_{string(ASCII)}$ est l'ensemble des chaînes de caractères composées de caractères ASCII.
- $\mathcal{L}_{string(8859-1)}$ est l'ensemble des chaînes de caractères composées de caractères définis dans la norme ISO-8859-1.
- $\mathcal{L}_{champs(ontologie)}$ est l'ensemble des noms de champ pour une ontologie donnée.

$$\mathcal{L}_{champs(DC)} = \{ \begin{array}{l} \text{“title”}, \\ \text{“creator”}, \\ \text{“subject”}, \\ \text{“description”}, \\ \text{“publisher”}, \\ \text{“contributor”}, \\ \text{“data”}, \\ \text{“type”}, \\ \text{“format”}, \\ \text{“identifier”}, \\ \text{“source”}, \\ \text{“language”}, \\ \text{“relations”}, \\ \text{“rights”}, \\ \text{“coverage”}\} \end{array}$$

est l'ensemble des quinze éléments dont la sémantique est définie par les workshops de *Metadata Dublin Core* ([WGMD95]).

$$\mathcal{L}_{Gemet1.5(fr)} = \{ \begin{array}{l} \text{“Agenda 21”}, \\ \text{“environnement”}, \\ \text{“pollution”}, \\ \ldots\} \end{array}$$

est l'ensemble des 5398 groupes nominaux français de la version 1.5 du thésaurus GEMET.

$$\mathcal{L}_{DDC-21} = \{ \begin{array}{l} \text{“000”}, \\ \text{“001”}, \\ \text{“002”}, \\ \ldots\} \end{array}$$

est l'ensemble des cotes de la $21^{ème}$ édition de la classification décimale de Dewey ([Bet93b]).

- $\mathcal{L}_{URL}$ est l'ensemble des URLs (*Uniform Resource Locators*) identifiant et localisant toute ressource sur le Web, et respectant la norme RFC 1738.
- $\mathcal{L}_{email}$ est l'ensemble de toutes les adresses électroniques qui répondent au RFC 720 *Address Specification Syntax for Network Mail.*

Definition 10 *Un schéma s est une fonction $s : \mathcal{A} \longrightarrow \mathcal{P}(\mathcal{V})$.*

Exemples :

Par exemple :

$$s((\text{“creator”}, \{(\text{“scheme”}, \text{“e-mail”})\})) = \mathcal{L}_{email}$$
$$s((\text{“creator”}, \phi)) = \mathcal{V}$$
$$s((\text{“subject”}, \{(\text{“scheme”}, \text{“Gemet1.5(fr)”})\})) = \mathcal{L}_{Gemet1.5(fr)}$$

Remarques :

Notons que la relation entre un document d et un ensemble de pages Web est matérialisée par le schéma :

$$s((\text{“}relation\text{''}, (\text{“}type\text{''}, \text{“}contains\text{''}))) = \mathcal{U}$$

Definition 11 *On dit qu'un attribut a est un attribut valide pour un schéma s, si et seulement si a appartient au domaine de définition de s.*

Definition 12 *On dit qu'un ensemble D de documents Web est compatible avec un schéma s si pour tout champ c d'un document $d \in D$ dont l'attribut est un attribut valide pour le schéma s, on a : $val(c) \subset s(attr(c))$*

Definition 13 *On appelle une collection de documents Web, un quadruplet ($\mathcal{N}$, $\mathcal{V}$, s, D) où :*

- *$\mathcal{N}$ est un espace de noms*
- *$\mathcal{V}$ est un espace de valeurs*
- *s est un schéma de $\mathcal{A} = \mathcal{N} \times \mathcal{P}(\mathcal{N} \times \mathcal{V})$ vers $\mathcal{P}(\mathcal{V})$*
- *D est un sous-ensemble de l'ensemble $\mathcal{D} = \mathcal{N} \times \mathcal{P}(\mathcal{A} \times \mathcal{V})$ de documents Web tel que :*
  - *D est compatible avec le schéma s,*
  - *id est une injection de D dans $\mathcal{N}$*

La condition d'injectivité de la fonction id de D dans $\mathcal{N}$ permet d'assurer un moyen d'identification des différents documents. La compatibilité de D avec le schéma s assure que tous les champs de tous les descripteurs de tous les documents ont des valeurs qui appartiennent aux domaines de valeurs (les langages) de leurs attributs respectifs.

Exemples :

- Considérons la collection $(\mathcal{N}, \mathcal{V}, s_0, D_0)$ où :
  s_0 est un schéma inspiré de Dublin Core :

$$(\text{“creator”}, \{(\text{“scheme”}, \text{“e-mail”})\})) = \mathcal{L}_{email}$$
$$(\text{“creator”}, \phi)) = \mathcal{V}$$
$$(\text{“creator”}, \{(\text{“scheme”}, \text{“Gemet1.5(fr)”})\})) = \mathcal{L}_{Gemet1.5(fr)}$$
$$(\text{“date”}, \{(\text{“scheme”}, \text{“ISO8601”})\})) = \mathcal{L}_{date(ISO8601)}$$
$$(\text{“date”}, \{(\text{“type”}, \text{“created”})\})) = \mathcal{L}_{date(ISO8601)}$$
$$(\text{“date”}, \{(\text{“type”}, \text{“modified”})\})) = \mathcal{L}_{date(ISO8601)}$$

-

Un site Web est un document Web particulier :

Definition 14 *Un site Web est caractérisé par Site(domain, {resource },[{WWWDocument }], [metadata.class]) where :*

- domain est le nom du serveur hébergeant les ressources Web (par exemple www.w3c.org) ;

- resource est n'importe quelle ressource disponible sur site Web à travers une URL ;

- WebDocument est une collection de pages ou ressources Web organisées en DAGs ;

- le fichier metadata.class est le descripteur de documents Web disponible sur un site Web, en général stocké à la racine du site et accessible via une URL ;

4.5 Les relations structurelles des documents Web

Nous allons maintenant considérer certains champs particuliers qui permettent d'induire des relations binaires internes dans une collection $(\mathcal{N}, \mathcal{V}, s, D)$ de documents Web.

Certaines de ces relations permettront de définir des structures, — typiquement des arbres, des graphes acycliques ou plus généralement des graphes, — dans l'espace des documents D. Ces structures seront utilisés pour la *propagation des attributs.*

Nous allons donner maintenant les définitions et les notations pour préciser ces notions.

4.5.1 Attribut relationnel

Definition 15 *Un attribut l est un attribut relationnel dans une collection $(\mathcal{N}, \mathcal{V}, s, D)$ de documents Web si pour tout champ d'attribut l, on a :*

$$val_l(D) \subset id(D)$$

Cela signifie que les valeurs des champs d'un attribut relationnel sont toutes des identifiants d'autres documents. Ainsi, on peut définir une relation binaire $\mathcal{R}_l$ dans D pour un attribut relationnel l.

Notations :

Pour tout attribut relationnel l, on définit la relation binaire $\mathcal{R}_l$ dans D, par :

$$(d_o, d_d) \in \mathcal{R}_l \quad \text{si et seulement si} \quad id(d_d) \in val_l(desc(d_o))$$

et on notera :

$$\mathcal{R}_l^0(d) = \{d\}$$

$$\mathcal{R}_l(d) = \{d_i \in D \mid (d, d_i) \in \mathcal{R}_l\}$$

et pour tout entier naturel n strictement positif :

$$\mathcal{R}_l^n(d) = \bigcup_{d_i \in \mathcal{R}_l^{n-1}(d)} \mathcal{R}_l(d_i)$$

et enfin :

$$\mathcal{R}_l^\infty(d) = \bigcup_{n>0} \mathcal{R}_n(d)$$

Ainsi :

$$\mathcal{R}_a^1(d) = \mathcal{R}_a(d)$$

Ces relations correspondent aux enfants, petits-enfants et descendants selon l'attribut relationnel l. À présent nous posons les notations pour les relations inverses des précédentes qui correspondent aux ascendants des documents Web :

$$\mathcal{R}_l^{-1}(d) = \{d_i \in D \mid (d_i, d) \in \mathcal{R}_l\}$$

et pour tout entier naturel n strictement positif :

$$\mathcal{R}_l^{-n}(d) = \bigcup_{d_i \in \mathcal{R}_l^{-n+1}(d)} \mathcal{R}_l^{-1}(d_i)$$

et enfin :

$$\mathcal{R}_l^{-\infty}(d) = \bigcup_{n<0} \mathcal{R}_l^{n}(d)$$

Exemples :

Pour l'ensemble D suivant :

$$D = \{d_0, d_1, d_2, d_3, \dots\}$$

avec :

d_0 = ("d0",
{
("child", ϕ, "d1"),
("child", ϕ, "d2"),
("child", ϕ, "d3"),
...
}
)

d_1 = ("d1", ...)
d_2 = ("d2", ...)
d_3 = ("d3", ...)
d_4 = ("d4", ...)

on a la relation suivante :

$d_1 \ \mathcal{R}_{child} \ d_0$
$d_2 \ \mathcal{R}_{child} \ d_0$
$d_3 \ \mathcal{R}_{child} \ d_0$

et on notera :

$\mathcal{R}_{child}(d) = \{d_1, d_2, d_3\}$	l'ensemble des enfants du document d,
$\mathcal{R}^2_{child}(d)$	l'ensemble des petits-enfants du document d,
$\mathcal{R}^{\infty}_{child}(d)$	l'ensemble des descendants du document d.

4.5.2 Propagation des attributs

Les valeurs des attributs des documents Web peuvent être propagées le long des liens de structure, ce que nous formalisons à l'aide des notations suivantes.

Soit a un attribut, l un attribut relationnel, et n un entier naturel, on note :

$$val_a^{l(n)}(d) = \bigcup_{d_i \in \mathcal{R}_l^n(d)} val_a(d_i)$$
$$\overline{val_a^{l(n)}}(d) = \bigcup_{m \leq n} val_a^{l(m)}(d)$$

4.6 Modélisation des requêtes

Nous abordons à présent le langage de requête associé à notre modèle de documents Web. Nous allons montrer qu'à travers ce langage, on peut aussi bien exprimer des requêtes similaires à celles des moteurs de recherche en texte intégral, que celle annuaire thématiques. Nous verrons également que ce langage permet de tirer partie de la structuration des documents Web en exprimant des requêtes faisant intervenir des graphes de documents Web. Les algorithmes mis en œuvre et le problème de complexité associé à ce langage sont par contre traités au chapitre implémentation. Nous aborderons cependant succinctement la technique algorithmique à la fin de ce chapitre.

4.6.1 Qu'est ce qu'une requête sur des documents Web ?

La plupart des systèmes de recherche d'informations classiques (SRI) proposent un système de recherche puissant sur le contenant d'un document, mais par contre, ils ne permettent pas de prendre en compte le contexte autour d'un document (c'est-à-dire les relations qui peuvent exister entre un document et d'autres dans le corpus). De fait l'utilisateur doit généralement lire chaque document trouvé et consulter les document annexes, puis reformuler sa requête. Ce problème induit une surcharge cognitive importante. Notre approche consiste donc a prendre en compte à la fois une recherche sur des champs d'un document Web et le contenu des ressources web qu'il référence et également à prendre en compte les liens structurant ces documents Web de manière à pouvoir formuler des questions qui prennent en compte le contexte autour d'un document Web.

Ainsi, dans le premier cas, il est possible d'envisager une requête recherchant tous les documents Web ayant pour champ SUBJECT "AGORA21" et qui, dans les pages qu'ils référencent (ici les ressources Web sont des pages HTML) contiennent le mot "environnement". Dans le second cas, un exemple serait de

rechercher un document WEB référençant des pages contenant le mot "jurisprudence", ce document devant être contenu (lié par un lien de type "CONTENT") dans un autre document référençant des pages contenant le mot "informatique". Avec un moteur de recherche classique on aurait pu formuler la requête sous la forme d'un combinaison logique des deux mots "jurisprudence AND informatique". Cependant cette approche ne fonctionne que si les deux mots sont contenus dans la même page HTML, alors qu'avec notre approche, si le document est réparti sur plusieurs pages HTML, notre approche permet de retrouver le document qui nous intéressent et de connaître l'ensemble des pages qui le forme.

Nous allons à présent spécifier formellement le langage de requête. Pour faciliter la lecture de ce formalisme, nous le présenterons en deux étapes : les requêtes simples, qui se focalisent sur un document WEB à rechercher, et les requêtes complexes qui prennent en compte la relations structurant un ensemble de documents WEB.

4.6.2 Les requêtes simples

Toute requête se rapporte à un document WEB ainsi le langage de requête début toujours par l'entête "select docweb" suivi des contrainte sur le document. Par exemple :

```
select docweb  begin
   field "subject" content "AGOR21" or "AGORA",
   field "author" with scheme "e-mail" content "agora@emse.fr",
   field "keyword" content "environment" and "water" and not "pollution",
   web resource content "water" or "liquid"
end select
```

Cette requête recherche un document web ayant un champ "subject" contenant la valeur "AGORA21" ou "AGORA", ayant également un champ "author" ayant pour qualificateur "e-mail" et pour valeur "agora@emse.fr", ayant un champ "keyword" contenant les mots clefs "environment" et "water" mais pas "pollution" et référençant une ressource web (page html) contenant le mot "water" ou "liquid"

La grammaire d'une requête simple s'exprime doc comme suit :

```
REQUETE           ::= select docweb begin  LISTE_CONTRAINTES end select ;
LISTE_CONTRAINTES ::= LISTE_CONTRAINTES "," CONTRAINTE  | CONTRAINTE ;
CONTRAINTE        ::= CONTRAINTE_SUR_CHAMP  | CONTRAINTE_SUR_RESSOURCE_WEB ;
```

```
CONSTRAINTE_SUR_CHAMP ::= field NOM_CHAMP LISTE_ATTRIBUTS content LISTE_VALEURS ;
VALEUR                ::= STRING
LISTE_ATTRIBUT        ::= with scheme VALEUR | ;
LISTE_VALEURS         ::= OPER VALEUR LISTE_VALEURS |
                          OPER VALEUR LISTE_VALEURS |
                          OPER VALEUR ;
OPER                  ::= OPER_BOOL | OPER_REL | ;
OPER_BOOL             ::= and | or | and not ;
OPER_REL              ::= ">" | "<" | ">=" | "<=" ;
CONTRAINTE_SUR_RESSOURCE_WEB ::= web resource LISTE_VALEURS ;
```

On note que le langage permet d'exprimer des opérations booléennes sur le contenu des champs et les mots des pages référencées par le document, mais également des opérateurs relationnels.

Ainsi on pourrait rechercher un document référençant un article technique publié entrent le mois de janvier et le mois de février 2000, ce qui s'exprime alors comme suit :

```
select docweb begin
   ...
   field "date" content >= "01/01/2000" and <= "29/02/2000",
   ...
end
```

Definition 16 *On appelle une requête simple un champ.*

Exemples :

Par exemple :

- (("creator", ϕ), "doan")
- (("creator", ("scheme", "e-mail")), "doan@emse.fr")

Definition 17 *Une fonction d'appariement est une fonction de* $\mathcal{V} \times \mathcal{V}$ *dans* $[0,1]$.

Exemples :

Nous avons par exemple :

- $f_{exact}(v_1, v_2)$ ssi $v_1 = v_2$
- $f_{regexp}(v_1, v_2)$ ssi v_1 s'apparie avec l'expression régulière v_2

Le but d'une requête est de trouver un ensemble de documents Web répondant à certaines caractéristiques exprimées dans la requête. Ces caractéristiques peuvent porter sur :

- les champs d'un document Web ;
- le contenu des ressources Web référencées par un document ;
- les relations de structure liant plusieurs documents.

Avant d'introduire les requêtes, nous devons nous doter d'outils mathématiques pour définir ces requêtes :

Une contrainte sur un champ d'un document est définie par le nuplet $co \in \mathcal{CO} = (d, a, t, v_1, ..., v_n) \in \mathcal{D} \times \mathcal{A} \times \mathcal{T} \times \mathcal{CV}^n$ où :

- $d \in \mathcal{D}$ est le document considéré.
- $a \in \mathcal{A}$ désigne un attribut d'un champ du document.
- $t \in \mathcal{T}$ désigne un type de contrainte.
- et $v_i \in \mathcal{V}$ est une valeur possible du champ.

Exemples :

Avec a et v se définissant comme pour les champs d'un document. Exemple :

- $(d_1, (\text{“author”}, \phi), =, \text{“doan”})$: pour le document d_1 et l'attribut “author” on définit la contrainte de type “=” (égalité) pour la valeur “doan”. c.-à-d que pour vérifier cette contrainte, le document d_1 doit contenir un champ d'attribut “author” ayant pour valeur “doan”.
- $(d_1, (\text{“author”}, \{(\text{“scheme”}, \text{“e-mail”})\}), =, \text{“doan”})$: idem que précédemment sauf que le l'attribut “author” doit avoir pour qualificateur “scheme=e-mail”.
- $(d_1, (\text{“date”}, \phi), in, \text{“10/12/1990”}, \text{“01/01/200”})$: dans ce cas précis la contrainte porte sur le le champ date du document d_1 qui doit être compris entre le 10/12/1990 et le 01/01/2000.

Comme nous utilisons un modèle vectoriel, nous ne voulons pas nous limiter à une valeur de contrainte VRAI ou FAUX. Nous définissons une fonction de similarité de l'ensemble des contraintes $co \in \mathcal{CO}$ vers l'intervalle réel $[0 \dots 1]$.

$$sim_t(co) \text{est une fonction telle que } sim_t : \mathcal{CO} \rightarrow [0 \dots 1]$$

la fonction retourne une valeur d'autant plus proche de 1 que la contrainte s'avère vérifié.

étant donné que nous utiliserons la logique des prédicats pour combiner les résultat, nous définissons également une fonction nous retournant un booléen VRAI (1) ou FAUX (0) déterminé à partir du résultat de la fonction de similarité

et d'un seuil $bool(rang, seuil)$ $bool : [0 \dots 1] \times [0 \dots 1] \rightarrow 0,1$ tel que

$$bool(rang, seuil) = \begin{cases} 0 & \text{si } rang < seuil \text{ et} \\ 1 & \text{si } rang \geq seuil. \end{cases}$$

Pour les besoins de notre application, nous avons défini un ensemble de fonctions de similarités minimales qui peuvent être étendues à volonté :

- $sim_{=}(co)$ avec $co = (d, a, =, v)$, retourne une valeur d'autant plus proche de 1 que le document considéré contient à pour valeur de l'attribut a une valeur proche de v.
- $sim_{<}(co), sim_{>}(co), sim_{\leq}(co), sim_{\geq}(co)$ avec $co = (d, a, \{<, >, \leq, \geq\}, v)$, retourne 1 respectivement si la valeur v de l'attribut a du document considéré est inférieur, supérieur, inférieur ou égale ou supérieur ou égal à la valeur réel du champ du document, 0 sinon
- $sim_{in}(co)$ avec $co = (d, a, in, v_1, v_2)$, retourne la valeur 1 si la valeur de l'attribut a du document considéré est compris entre v_1 et v_2
- $sim_{contains}(co)$ avec $co = (d, a, contain, v)$, retourne une valeur proche de 1 si l'attribut a est une référence sur une page du document et que la valeur v apparaît dans la page référencée par l'attribut a.

Nous pouvons à présent définir une requête $r \in \mathcal{R}$ simple sur un document comme expression booléenne combinant des opérateurs booléen $\wedge, \vee$ ou $\neg$, comme suit :

$$r = [\neg]co_1 op_1 [\neg]co_2 op_2 \dots op_n [\neg]co_n$$

où $co_i \in \mathcal{CO}$ est une contrainte et $op_i \in \{\wedge, \vee\}$ est un opérateur booléen de conjonction (ET) ou de disjonction (OU) et $[\neg]$ indique un opérateur facultatif de négation.

Le but d'une requête est de restituer les documents répondant à des contraintes précises, nous définissons la fonction req comme suit :

$$req(r, s_1, ..., s_n) : \mathcal{R} \times [0 \dots 1]^n \rightarrow \mathcal{D} \text{ où}$$
$$req(r, s_1, ..., s_n) = \{d \in \mathcal{D} / bool(r, s_1, ..., s_n) = 1\}$$

la fonction $bool$ est une extension de la fonction précédente mais appliqué à une requête est non plus une simple contrainte. Elle se construit comme suit :

- remplacer chaque occurrence de co_i dans la requête par $bool(sim(co_i), s_i)$
- $bool(r, s_1, \ldots s_k)$ est vraie uniquement si l'expression booléenne exprimée dans r est vraie en prenant en considération les seuils de similarités $s_1, ..., s_n$.
- la requête retourne donc l'ensemble des documents dont la fonction *bool* retourne la valeur vraie.

L'intérêt de préciser un seuil ici, permet de changer dynamiquement lors de la requête l'importance d'une contrainte. Ainsi si l'on recherche un article, on pourra donner un seuil très grand pour une date supérieure à janvier 2000 par exemple et ne pas garder la même importance à chaque contrainte.

Il est également intéressant de pouvoir calculer une fonction de similarité pour un document retrouvé qui peut être par exemple une simple moyenne des fonctions de similarités de chaque contrainte : $sim(d) : \mathcal{D} \rightarrow [0 \ldots 1] = \frac{\sum_{i=1}^{n} sim2(co_i)}{n}$ ou $sim2(co_i) = sim(co_i)$ si la contrainte n'est pas précédée de l'opérateur de négation et $sim2(co_i) = 1 - sim(co_i)$ si la contrainte est précédée de l'opérateur de négation ($\neg$) dans l'expression de la requête.

4.6.3 Les requêtes sur la structure

S'il est intéressant de pouvoir retrouver un document ayant certaines caractéristiques, la force de notre modèle réside dans la possibilité de prendre en compte le contexte, c'est-à-dire retrouver des documents liés à d'autres par certaines caractéristiques.

Une telle contrainte dite relation se décrit par le nuplet $co_{rel} = (d_1, d_2, \mathcal{R}_t, card)$ qui exprime le fait que le document $d_1 \in \mathcal{D}$ doit être lié au document $d_2 \in \mathcal{D}$ par une relation $\mathcal{R}_t$ de cardinalité maximale *card.* ce qu'on l'on a noté précédemment dans la section "relations structurelles des documents Web" par $d_1\ mathcal R_t^{card}\ d_2$. On note $\mathcal{CO}_{\nabla|\updownarrow}$ l'ensemble des contraintes relationnelles possibles.

De même que pour les contraintes sur un document, nous définissons des fonctions de similarité et une fonction booléenne comme suit :

$sim_{rel}(co_{rel}) : \mathcal{CO}_{\nabla|\updownarrow} \rightarrow [0 \ldots 1]$

et

$bool(rank, seuil) : [0 \ldots 1] \times [0 \ldots 1] \rightarrow \{0, 1\}$ une fonction qui à partir d'une information de similarité est d'un seuil retourne la valeur 1 ou 0 selon que la similitude est supérieure ou inférieur au seuil donné.

À partir de là, une requête faisant intervenir la structure des documents consiste à retrouver un nuplet de documents $(d_1, d_2, \ldots, d_n)$ issus de requêtes

simples tel que ces documents soit liés par des contraintes relationnelles $(co_{rel_1}, co_{rel_2}, \ldots co_{rel_m})$ comme suit :

$r_{rel} = ((d_1, d_2, \ldots d_n), (co_{rel_1}, co_{rel_2}, \ldots co_{rel_m}))/i = 1 \ldots n, d_i \in req(r_i, s_1, \ldots, s_k)$ et $\forall co_{rel_i} = (d_{i_1}, d_{i_2}, t, card)$ alors $(d_{i_1}, d_{i_2}) \in \{d_1, \ldots d_n\}^2$ et $bool(sim_{rel}(co_{rel_i}, s_i) = 1$.

4.7 Application : un système de recherche d'information spécialisé dans un domaine

Dans cette section, nous validons les concepts que nous avons défini dans ce chapitre en proposant une application concrète qui repose sur deux aspects. Le premier expérimente la création des méta-informations sur un site Web, [DBGJ99]. Le deuxième point est la création d'un système de recherche d'information pour retrouver les données pertinentes sur ce site.

4.7.1 Création des méta-informations

Pour créer la méta-information associée à une organisation hiérarchique de documents Web, nous nous intéressons aux points suivants :

- Le site `http://www.agora21.org` spécialisé dans le domaine environnement et entreprise sera considéré comme une base de donnée test pour implanter la méta-information.
- un thésaurus sur le domaine de l'environnement (CREDOC ou autre) nous servira à indexer les documents. D'autre part, nous adopterons une classification normalisée type CDD.
- Pour formaliser les méta-information, nous utiliserons un format standard XML.
- La sémantique des méta-information associées aux documents sera décrite dans un dictionnaire de données.
- Nous décrirons dans quelle mesure une automatisation de la création des méta-informations pourra aider le créateur du site à fournir ces dernières.
- Le problème lié à la maintenance des méta-information sera évoqué.

Dans cette partie création du fichier metadata.class, deux solutions sont possibles :

- La première solution consiste pour l'auteur à ajouter des balises META à l'intérieur des pages HTML afin d'indiquer qu'elles appartiennent à un ou plusieurs documents Web. Ces documents Web sont hébergés sur le serveur HTTP dans le fichier metadata.class.
  Par exemple :

  ```
  <META NAME="containedIn"
  CONTENT="http://groseille.emse.fr/metadata.class#d0>
  ```

  Cette solution est avantageuse pour l'administrateur des documents Web car lorsqu'une page est supprimée ou modifiée, il est peu ou pas du tout besoin de changer les documents Web. Si les auteurs des pages facilitent la description sémantique de celles-ci en ajoutant des balises META, alors il est tout à fait concevable d'utiliser un outil de regroupement de pages ou de catégorisation par concepts pour construire les documents Web automatiquement.
- Une autre solution consiste à décrire totalement les documents Web en spécifiant quelles pages leur appartiennent.
  Par exemple :

  ```
  <DOCUMENT>
    <DC:Relation>
          <type="contains">
          "http://www.emse.fr/~brodhag/projelev/RESOURCES/index.html"
          </type>
      </DC:Relation>
  </DOCUMENT>
  ```

  Cette solution a l'avantage de permettre la création des méta-informations par une entité externe, par exemple un logiciel automatique, un expert ou un spécialiste qui n'est pas forcément l'auteur des pages Web. De plus, un même site Web pourra être décrits de plusieurs façons différentes, les documents Web correspondent alors à autant de vues différentes de ce site.

À partir de là, les documents peuvent être générés manuellement ou semi-automatiquement :

- manuellement, l'auteur des pages du site crée chaque document en le décrivant par des mots du thésaurus et en l'attachant à des documents et à un ensemble de pages par des relations sémantiques (par exemple liens de spécialisation ou de généralisation). Le formalisme standard uti-

lisé pour construire un document est spécifié plus loin.

- Un algorithme de "clusterisation" utilisant le filtrage et le mapping peut être employé afin regrouper un ensemble de pages qui concernent les nœuds de la CDD ou donnés par le thésaurus. Il s'agit ici d'indexer des "clusters" de pages et d'associer à chaque nœud de l'arbre du thésaurus les documents construits.

Pour notre expérimentation, un outil d'édition de documents Web a été développé au sein de l'équipe cf. Annexe C, et c'est un documentaliste qui s'est chargé de la création des méta-informations pour le site `http://www.agora21.org`. Cet outil utilise le thésaurus en ligne GEMET1.5. Les documents Web ont été sémantiquement décrits selon le thésaurus GEMET1.5 et la classification décimale de Dewey.

4.7.2 Maintenance des méta-informations

La maintenance est soit manuelle soit automatique selon que les documents Web ont été créés manuellement ou automatiquement.

4.7.3 Indexation des méta-informations

Dans cette section, nous expliquons nos choix d'implémentation dans la construction de l'index du site Web, [DBGJ98].

Les pages du site environnement sont tout d'abord indexées par un moteur de recherche local (par exemple Harvest), de même que la méta-information. Un spécialiste est créé selon une demande. Dans notre exemple, le spécialiste "environnement" va stocker dans sa base locale les pages et les documents Web indexés.

Lors de la phase d'indexation d'un site Web, nous gardons la totalité du contenu des documents Web ainsi que les méta-informations et la structure des documents Web.

Afin d'optimiser l'algorithme de recherche dans le DAG des documents Web, nous propageons les attributs dynamiques des documents Web au moment de l'indexation. L'index est donc construit a priori.

Les méta-informations qui sont des informations structurées sont stockées dans une base de données spécifique alors que le contenu des pages est sauvé dans un système de fichiers.

Le site environnement sera indexé par un robot spécialiste dont le rôle est de transmettre ou échanger la méta-information aux spécialistes ou généralistes

s'intéressant à son domaine de spécialisation. Nous nous intéressons à spécifier les points suivants concernant le robot spécialiste :

- Définir son domaine d'activité ou de spécialisation.
- Définir sa stratégie de recherche d'information et d'indexation.
- Définir son fichier de configuration qui détermine son profil (intérêts et compétences), la connaissance qu'il a de ses voisins, son protocole (formalisme des requêtes, interactions avec les autres robots), ainsi que la maintenance de son index de méta-information (par exemple, fréquence de mise à jour).
- Définir un algorithme pour "clusteriser" les documents et répondre aux requêtes des utilisateurs ou de ses voisins.

4.7.4 Interface du système de recherche d'informations

Le processus de recherche est basé sur la hiérarchie de contextes (cf. figure fig.4.4), ainsi que sur la propagation des attributs externes dynamiques le long de cette hiérarchie. Cette recherche est combinée avec une recherche textuelle sur le contenu des documents Web.

Nous pouvons désormais poser plusieurs types de requêtes, qui mettent en évidence l'intérêt des documents Web par rapport à une recherche d'informations classique sur le Web.

Requêtes contextuelles

Q = subject :pollution AND text :(nitrate or fluor)

Requêtes structurelles et textuelles

Q = subject :enterprise AND subject :pollution AND text :"problème d'hygiène"

Présentation des résultats

Les résultats (cf. fig. 4.5 sont présentés dans leur contexte, ce qui permet à l'utilisateur de synthétiser plus rapidement les réponses.

4.7.5 Conclusions

Un moteur de recherche classique (Alta Vista, lycos) indexe le Web en extrayant les mots du texte de chaque page téléchargée par un robot.

Notre approche consiste d'une part à indexer le Web en extrayant des informations des documents plutôt que les pages simples, d'autre part à extraire l'organisation hiérarchique d'un site. Nous voyons deux avantages à cette approche :

- les informations sont créées et contrôlées sur un document par le créateur du site ou un expert grâce à des méta-informations. Ceci implique plus de richesse sémantique et plus de structuration de l'information,
- l'organisation hiérarchique d'un site facilite la recherche exploratoire et favorise une explication de contexte.

Dans la suite de ce travail, la génération automatique de documents Web a été envisagée, validée dans le cadre d'une thèse en cours au département RIM de l'ENSMSE. La méthode utilisée consiste à appliquer des techniques de clusterisation de systèmes hypertextes dans des sites Web, afin de générer un arbre de contexte de pages Web [ADB99c],[ADB99b]. Il existe deux niveaux d'index : le premier niveau correspond au contenu des nœuds atomiques qui sont les pages Web, le deuxième niveau repose sur un ensemble de nœuds regroupés par une fonction de similarité et qui forment les clusters de pages Web. Les clusters sont organisés selon une hiérarchie sensée représenter un contexte lié à des pages Web induit par l'ensemble des pages d'un même site Web. L'utilisateur final peut ainsi interroger le contenu des pages Web en précisant un certain contexte.

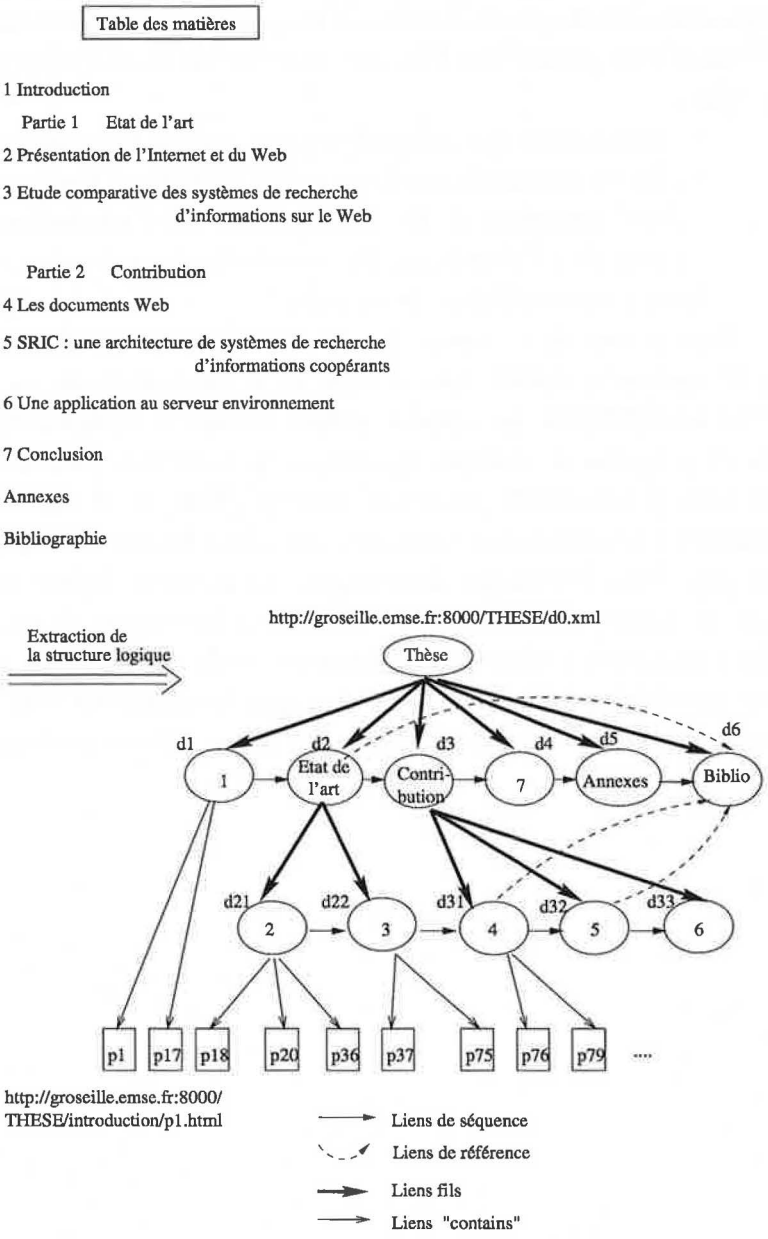


FIG. 4.1 – Organisation des document Web d'une thèse

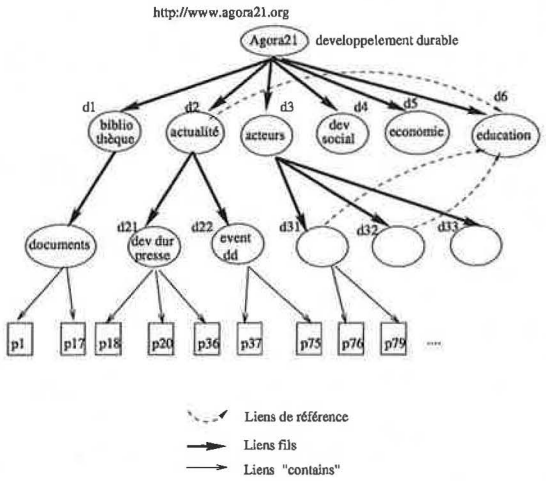


FIG. 4.2 – Organisation des document Web du site Agora21

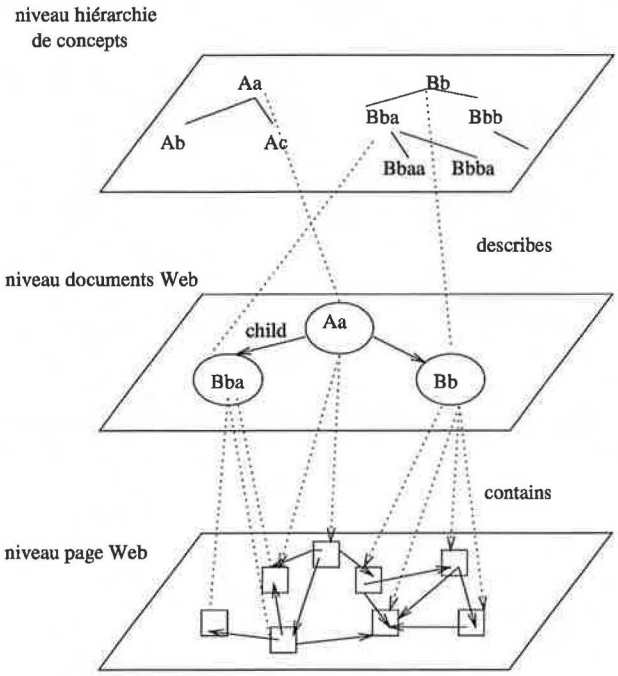


FIG. 4.3 – Architecture des documents Web

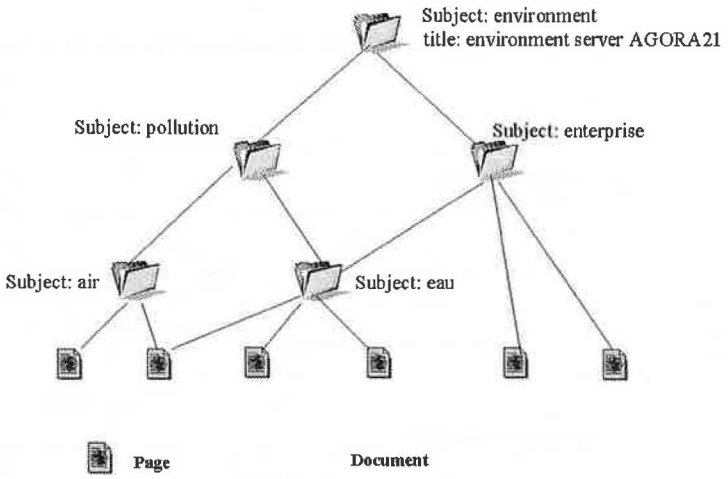


FIG. 4.4 – Hiérarchie des documents Web et leurs contextes

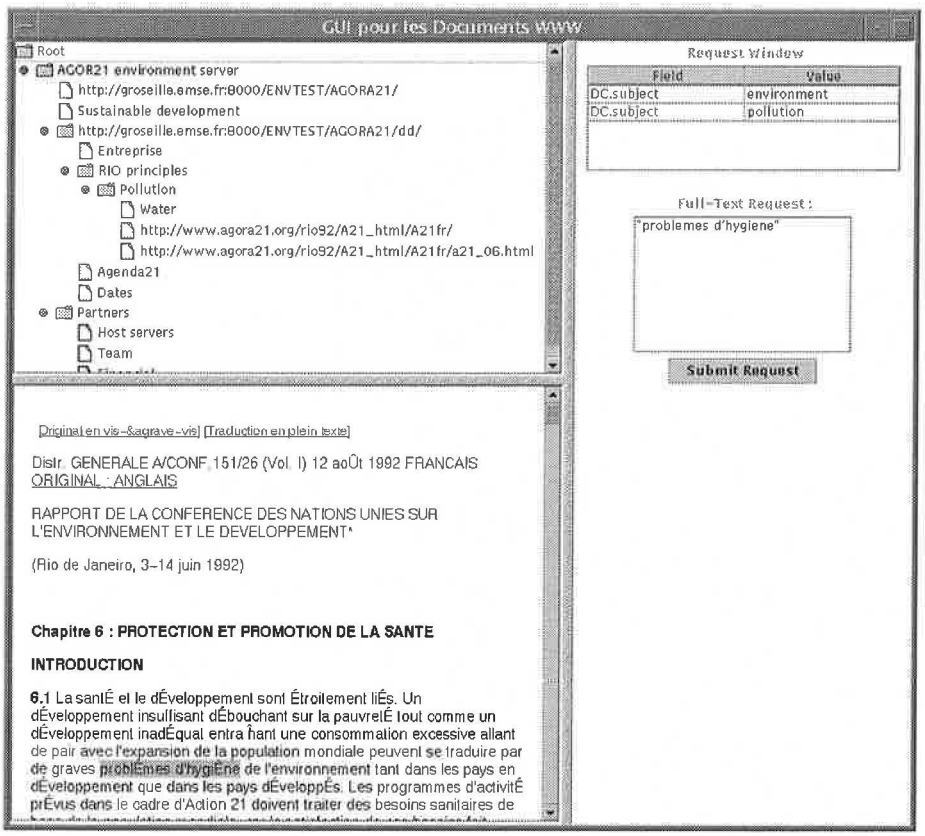


FIG. 4.5 – Interface utilisateur d'un SRI sur Internet

Chapitre 5

SRIC : Une architecture de systèmes de recherche d'informations coopérants

5.1 Introduction

Nous venons de voir dans le chapitre précédent l'intérêt de la structure et des méta-informations pour améliorer la qualité de description d'un site WWW. Nous avons également décrit un nouveau système de recherche d'information qui prendrait en compte ces éléments d'information fournis par l'auteur ou à l'aide d'outils d'indexation automatique.

Cette solution permet de réduire le bruit et le silence dans le cas d'une portion réduite du WWW, le plus souvent limitée à un site WWW. Afin d'étendre cette solution à l'ensemble du WWW, nous introduisons le concept de système de recherche d'informations coopérant. Nous définissons deux types de systèmes de recherche d'informations coopérants, qui diffèrent par leurs fonctionnalités : les spécialistes et les généralistes. Ce chapitre s'articule ainsi : tout d'abord nous définissons le concept général des spécialistes et des généralistes. Puis nous détaillons les fonctionnalités communes et spécifiques à chaque type d'outil. À partir de là, nous organisons les différentes entités coopérantes qui participent à l'architecture globale, d'une part en définissant les liens qui les structurent entre elles, d'autre part en explicitant leur rôle, c'est-à-dire en spécifiant les

contraintes, les ressources et les interactions au moment de la requête.

5.1.1 Vers une architecture coopérante

Le WWW est un système spatialement réparti et en constante évolution. De par sa nature, deux problèmes sont omniprésents lors de la mise en œuvre des moteurs de recherche classiques :

- la croissance exponentielle du volume des pages du WWW.
- la volativité des pages et la fréquence de leur mise-à-jour.

Pour résoudre le problème de volatilité des données, le moteur doit veiller à ce que son volume de données, qu'il s'agisse de méta-informations ou de données brutes ne dépasse pas un certain seuil afin d'assurer une fréquence de mise-à-jour et de collecte d'information raisonnable. De plus, étant donnée l'impossibilité d'indexer tout le WWW par un seul moteur, il est raisonnable de répartir cet aspect sur un ensemble de moteurs spécialisés se focalisant chacun sur des domaines plus ou moins complémentaires.

Ces considérations nous amènent tout naturellement à envisager la recherche de l'information sur le WWW comme un système d'outils coopérants que nous avons dénommés SRICs. Ainsi l'élément principal de cette architecture est un outil de recherche spécialisé dans un domaine que nous nommerons dans la suite de ce rapport *spécialiste*.

- Les spécialistes correspondent au type de système de recherche d'informations illustré au chapitre précédent. Ils n'indexent que les documents et les pages associées à leur spécialité. Un spécialiste est créé à la demande. Les spécialistes utilisent les méta-informations pour décider si oui ou non et à quel niveau de détail ils vont indexer la hiérarchie de documents ainsi que leurs pages. Ainsi il est possible de créer également une échelle de spécialistes, du très spécialisé (il connaît le domaine de façon très pointue), au moins spécialisé (connaissance horizontale du domaine).

Cette architecture de spécialiste nous amène à un nouveau problème. Il s'agit de disposer de la liste des moteurs pour qu'un utilisateur puisse choisir le spécialiste adéquat, ou bien par exemple pour qu'un spécialiste route une requête vers un outil d'un sous-domaine plus pointu. Ce qui nous amène à intégrer au SRIC une nouvelle entité : *l'annuaire*.

- Les annuaires sont exclusivement réservés à l'identification des SRICs et à leur référencement. Ils stockent peu d'informations afin de rendre leur accès rapide et efficace. Les annuaires sont des entités qui doivent être

connues et qui peuvent être organisées hiérarchiquement selon la localisation géographique (cf. annuaire distribué).

En dernier lieu se pose le problèmes des requêtes faisant intervenir des domaines de compétence de spécialistes différents. Il faut disposer d'un outil disposant d'une connaissance des domaines des différents spécialistes sans pour autant retomber dans le problème des outils classiques qui tentent d'indexer tout le WWW. C'est le rôle des *généralistes* qui n'indexent que les méta-informations et peuvent donc avoir une base de données beaucoup plus large qu'un spécialiste. Lorsque cela s'avère nécessaire, le généraliste contacte le ou les spécialistes les plus à même de donner une réponse à une requête donnée faisant intervenir plusieurs domaines de compétence, réalise la synthèse ou fusion des résultats et renvoie la réponse à l'utilisateur.

- Les généralistes ajoutent ou fusionnent les méta-informations à leurs index plutôt que d'indexer tout le WWW et les sous-arbres des documents. Ils n'ont pas de domaines de connaissance particuliers, ils se contentent de collecter des méta-informations sur des sites WWW ou sur des services comme d'autres spécialistes ou généralistes. Les généralistes sont censés avoir une vue globale et synthétique de l'ensemble du WWW (sans distinction du domaine d'intérêt), mais peuvent avoir une couverture limitée à une zone géographique par exemple, ou bien une couverture linguistique. Ceci implique la possibilité de hiérarchiser les généralistes. Comme nous l'avons vu, le principal but des généraliste est de faire le lien entre spécialistes de domaines différents.

On notera que les fonctionnalités des généralistes et des spécialistes diffèrent peu et qu'il s'agit surtout d'une différence conceptuelle sur la globalité des domaines de connaissance. Le protocole que nous définissons dans ce document est utilisé par ces différentes entités afin d'échanger des index, que nous appelons méta-informations ou descripteurs de documents, dans le but de faciliter une auto-organisation de l'information. Cette auto-organisation est une notion qui traduit une organisation dynamique et progressive de l'information, mise en place grâce à notre architecture que nous allons préciser par la suite.

5.1.2 L'organisation des SRICs

Nous allons aborder à présent le problème de l'organisation des spécialistes. L'architecture coopérante tend à résoudre le problème de la croissance expon-

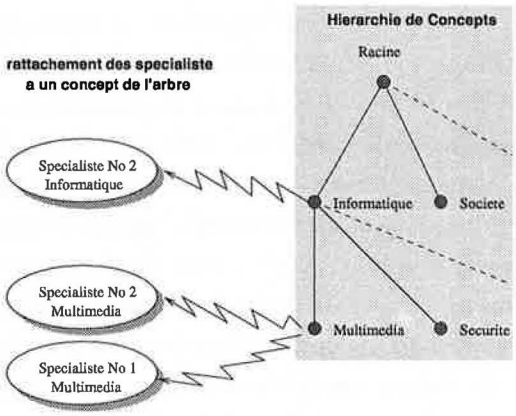


FIG. 5.1 – Rattachement des spécialistes à la hiérarchie de concepts

tielle du web et la volatilité des pages en distribuant les tâches d'indexation et de maintenance des données sur les différents spécialistes. Cependant elle introduit une nouvelle problématique, à savoir comment retrouver le(s) SRIC(s) pertinent(s) par rapport à une requête donnée.

L'approche suivie dans le cadre de cette thèse consiste à structurer ces SRICs autour d'une hiérarchie de thèmes. Cette hiérarchie est définie au travers d'un thésaurus ou d'une classification de concepts plus générale. Cela peut être par exemple la classification Dewey ou bien une hiérarchie de concepts ad-hoc telle celle de yahoo. Rien ne nous empêche également de structurer ces mêmes SRICs selon plusieurs hiérarchies, cependant pour simplifier l'exposé suivant, nous nous limiterons à des exemples basés sur une seule hiérarchie. Chaque spécialiste est rattaché (cf. FIG. 5.1) à un concept de cet arbre (son domaine de spécialisation). Lors d'une requête, le contexte ou le sujet de la requête permet de déterminer la liste des SRICs à même de répondre au mieux à la recherche d'information demandée.

Ce qui nous amène tout naturellement à la notion d'annuaire. Un annuaire a pour rôle d'identifier les SRIC relevant d'un domaine précis et de retourner des pointeurs sur ces SRICs. Pour éviter de retomber dans les problèmes de "*scalabilité*", l'architecture de ces annuaires est organisée selon un schéma semblable à celui utilisé pour les serveurs de noms de domaines des DNS. Cette architecture qui a déjà prouvé sont efficacité est reproduite dans le cadre des annuaires de SRICs avec cependant les nuances suivantes :

– l'arborescence traitée ici concerne la hiérarchie des thèmes (concepts) selon

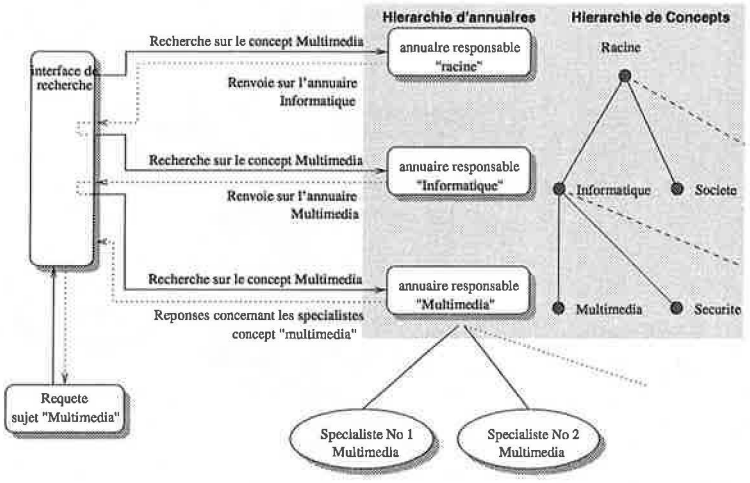


FIG. 5.2 – identification des SRICs pertinents pour une requête donnée

laquelle sont organisés les SRICs et non pas les noms de domaines utilisés pour identifier les machines du web ;

- l'information retournée dans notre cas sont des pointeurs vers les SRICs rattachés à un nœud (thème) de l'arbre et non pas l'adresse IP associée à un nom de domaine ;

Le schéma de la figure illustre le processus permettant d'identifier les SRICs pertinents pour un contexte donné lors d'une requête.

Un point important ici est que chaque annuaire a une connaissance totale de l'arbre des concepts qui structure nos SRICS. Par contre il ne connaît que les SRICS rattachés aux nœuds dont il a la responsabilité ainsi que les annuaires directement responsables des sous-nœuds de sa zone.

Cette distinction est importante, car l'annuaire pour nous renvoyer vers un autre annuaire devra être capable à partir d'un concept de connaitre sa position dans la hiérarchie par rapport aux nœuds dont il est responsable. Cette contrainte ne pose cependant pas de problème d'implémentation car contrairement aux services DNS, ce n'est pas l'arbre qui peut croitre de manière exponentielle et qui est inconnu à l'avance, mais les SRICs rattachés aux nœuds de cet arbre.

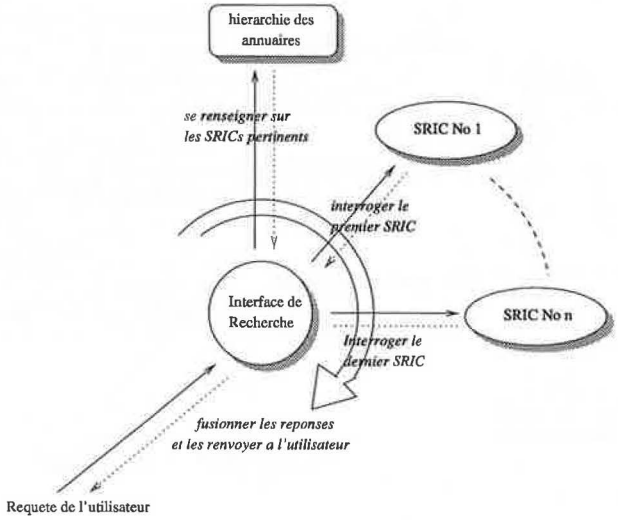


FIG. 5.3 – Processus de résolution d'une requête

5.1.3 Interface de recherche

Cette démarche nous amène à introduire un nouvel élément dans l'organisation des SRICs : l'interface de recherche. Le processus de recherche est une tâche complexe, en effet, elle implique de trouver les SRICs pertinents par rapport à une requête, de les interroger et éventuellement de fusionner leurs réponses.

Il est donc conceptuellement plus pertinent de séparer le module de recherche des SRICs spécialistes et généralistes. La vocation d'un spécaliste ou d'un généraliste étant de répondre à une question relevant de son domaine de compétence, l'interface a par contre pour but de prendre en compte la nature distribuée de ces SRICs et de prendre en charge tout le processus de propagation de la requête de la récupération et de la présentation des résultats.

Le schéma de la figure explicite ce processus. Lors d'une requête de l'utilisateur, l'interface de recherche va interroger les annuaires selon le mode vu à la section précédente. Une fois déterminé les SRICs pertinents, il interrogera chacun de ces SRICs à tour de rôle, fusionnera les réponses, puis retournera les résultats ainsi synthétisés à l'utilisateur.

Par la suite, nous considérerons que chaque SRICs dispose d'une interface de recherche capable de prendre en charge tout ce processus. Cependant rien n'empêche la mise en place de telles interfaces indépendamment de tous SRICs, le cas des méta-moteurs de recherche tel metacrawler en est un exemple.

5.1.4 Recherche multi-thèmes

Jusqu'à présent, nous avons uniquement considéré des recherches concernant un thème spécifique pour lequel il existe un spécialiste compétant. Qu'en est il du cas où une recherche concerne deux thèmes différents à la fois ? Par exemple en supposant que nous recherchons des documents se rapportant à la jurisprudence des infractions dans le domaine informatique. Ces informations relèvent à la fois de compétences en informatique ainsi que de compétences juridiques.

Il s'avère possible de créer un spécialiste ayant des compétences dans ces deux domaines, cependant cette approche compliquerait à la fois la gestion de la base des annuaires et supprimerait l'organisation hiérarchique de nos spécialistes. Or comme nous l'avons indiqué précédemment, la structuration des documents est intimement liée à sa nature hiérarchique.

L'approche que nous avons suivie, consiste à définir une nouvelle catégorie de SRICs dénommée *généralistes*. Les *généralistes* n'indexent que les méta-informations et peuvent donc avoir une base de données beaucoup plus large qu'un spécialiste tout en évitant les problèmes de scalabilité.

La figure 5.4 explicite le processus. Lorsqu'un spécialiste indexe des documents multi-thèmes il ne conserve que les méta-informations propres à sa thématique, et propage l'information (c'est à dire les documents multi-thèmes) vers un généraliste adéquat. De plus le spécialiste indexe également les pages référencées par ces documents ce qui lui permet de combiner une recherche sur les méta-informations avec la recherche plein texte des moteurs classiques. Pour sa part le généraliste indexe également les documents multi-thèmes cités ci-dessus mais conserve toute l'information relative aux différents thèmes et garde en plus une référence sur le spécialiste ayant indexé ces documents. Ainsi le généraliste peut répondre à une question relative aux méta-informations de ces document ou propager la requête vers le(s) spécialiste(s) référencé(s) lorsque la requête porte également sur le contenu de la page.

La question cruciale dans notre cas est de déterminer le généraliste adéquat. C'est-à-dire comment retrouver de manière univoque et sans recherche exhaustive à travers tous les généralistes existants celui (ou ceux) qui à (ont) potentiellement indexé les documents relevant d'un domaine multi-thèmes.

Comme nous l'avons vue précédemment lors d'une requête par l'interface de recherche le système d'annuaire permet de retrouver le(s) spécialiste(s) du domaine en fonction du thème de la requête. Lorsque la requête porte à la fois sur plusieurs thèmes l'approche est la même mais avec les différences suivantes :

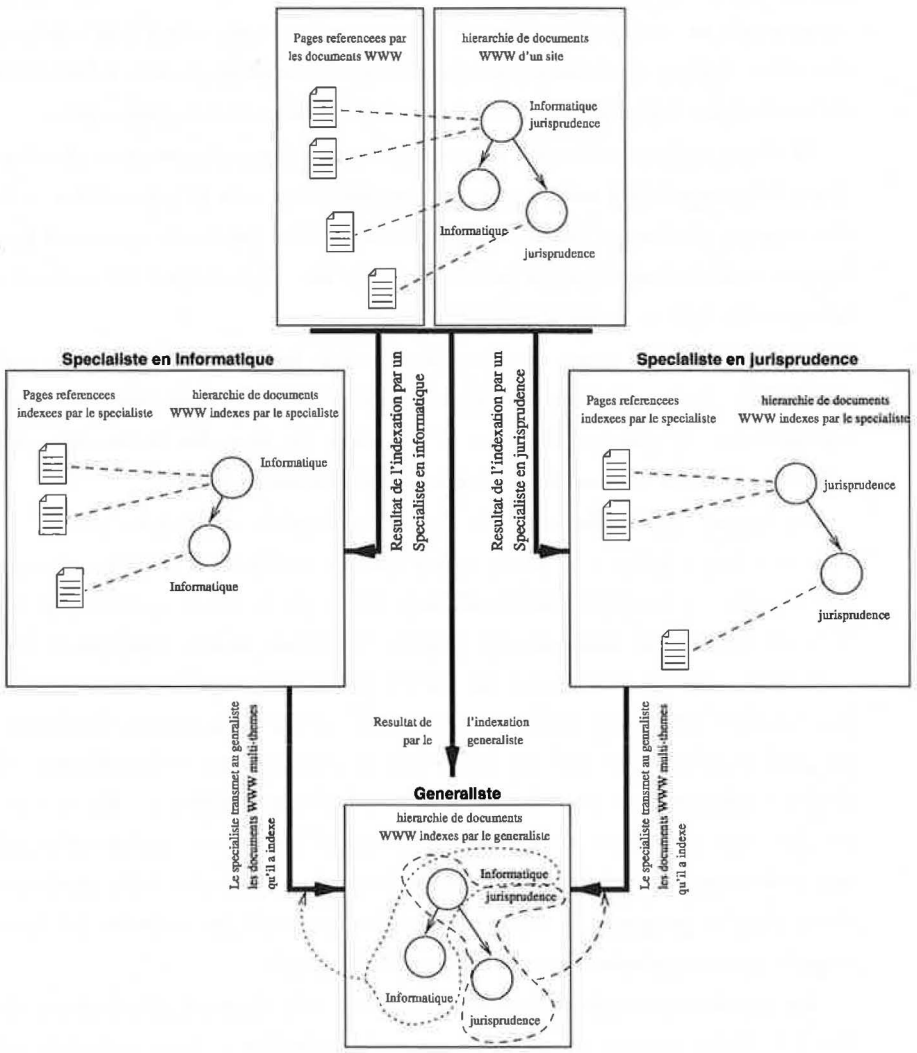


FIG. 5.4 – Mécanismes d'indexation de documents multi-thèmes

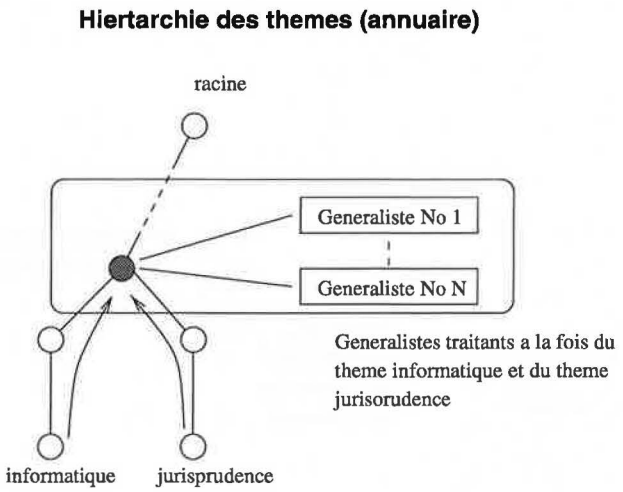


FIG. 5.5 – Retrouver un généraliste responsable de plusieurs thèmes

- chaque nœud de l'arbre de concepts (thèmes) utilisé pour hiérarchiser les spécialistes sert également à référencer les généralistes.
- le(s) généraliste(s) qui assure(nt) (cf. figure 5.5) la collecte d'information concernant n thèmes differents est(sont) celui(ceux) situé(s) au nœud le plus éloigné de la racine tel qu'à partir de cette même racine, il faut obligatoirement passer par ce nœud avant d'aboutir à chacun des nœuds correspondant aux n thèmes cités.

Un spécialiste qui indexe des documents multi-thèmes supprime ce qui se rapporte à d'autres thèmes mais propage l'information aux généralistes identifiés ci-dessus. Le généraliste n'indexe que les documents mais prend en compte toutes les meta-informations relatives à ce document (donc tous les thèmes) et garde trace du spécialiste ayant indexé ces méta-informations. Lors d'une recherche multi-thèmes, on contacte le généraliste qui renvoie les questions relatives aux pages aux spécialistes des domaines mais peut quant à lui répondre aux questions concernant uniquement les méta-informations.

5.1.5 Vue globale de l'architecture

Nous pouvons détailler à présent l'architecture globale de notre système de recherche coopérants (cf. figure 5.6).

1. lors d'une recherche, l'interface de recherche va questionner l'annuaire pour obtenir la liste des spécialistes pertinents lorsqu'il s'agit d'une requête à

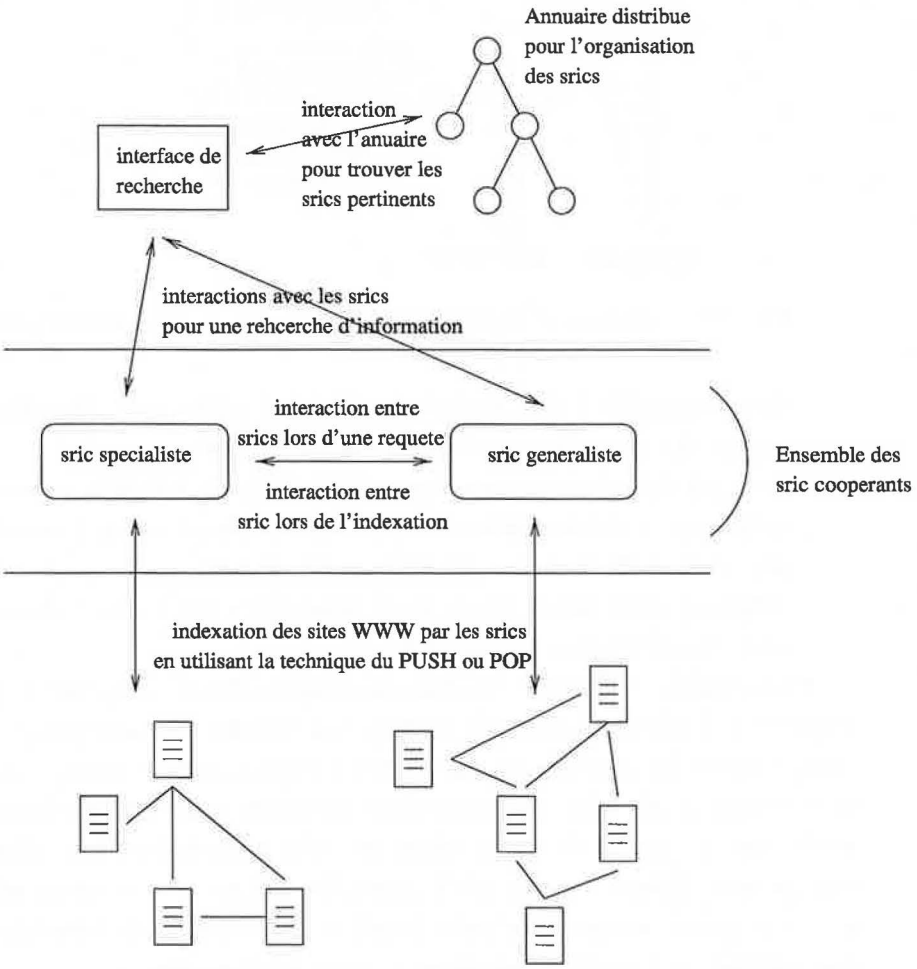


FIG. 5.6 – Architecture globale des SRICs

thème unique, où bien la liste des généralistes pertinents lorsqu'il s'agit d'une requête multi-thèmes.

2. dans une seconde étape, l'interface de recherche va questionner directement les spécialistes ou généralistes pour obtenir une reponse à la requête de l'utilisateur :
   - lorsqu'il sagit d'une requête mono-thème, elle propage la requête à tous les spécialistes pertinents, puis fusionne les résultats obtenus avant de la présenter à l'utilisateur.
   - lorsqu'il s'agit d'une requête multi-thèmes, elle propage la requête vers les généralistes pertinents. Ceux-ci lui répondent directement lorsque la requête ne porte que sur les méta-informations ou propagent à leur tour la requête vers les spécialistes qu'ils ont référencés concernant ces méta-informations pour prendre en compte les critères d'indexation spécifiques aux pages.

   Une fois les réponses obtenues, la fusion s'effectue en supprimant les doublons des réponses et classant ces résulats.
3. les SRICs indexent pour leur part les sites WWW en récupérant les méta-informations de ces sites par la technique du PUSH ou du PULL et en indexant en plus les pages référencées par ces méta-informations dans le cas des spécialistes. Les spécialistes et généralistes peuvent interagir à ce niveau dans deux cas :
   - les spécialistes propagent les méta-informations vers le(s) généraliste(s) adéquat(s) lorsque les méta-informations qu'ils indexent portent sur d'autres thèmes en plus des leurs.
   - les généralistes propagent les méta-informations qu'ils indexent vers les spécialistes qui relèvent des thèmes de ces méta-informations.

Nous allons à présent détailler les différents éléments du système puis dans un deuxième temps nous expliciterons les protocoles d'interaction mis en œuvre entre les SRICS.

5.2 Les entités du système

5.2.1 Les sites WWW

Les sites WWW dotés de méta-informations participent à l'alimentation en ressources du système global soit par la technique du push (un site WWW

envoie lui-même une description de son contenu à un ou plusieurs services, lors de la création du site ou lors de ses mises à jour), soit par la technique du pull (les méta-informations sont collectées par les services du système à une certaine fréquence, mais cela pose le problème d'asynchronisme). Cette description est accessible à travers le fichier metadata.txt situé à la racine et décrivant aussi bien les documents WWW du site que les services (spécialistes et généralistes) disponibles sur le site. Ce fichier est rempli par l'administrateur du site.

5.2.2 Les annuaires

Les annuaires sont exclusivement réservés à l'identification des SRICs et à leur référencement. Ils stockent peu d'informations afin de rendre leur accès rapide et efficace. Les annuaires sont des entités qui doivent être connues et qui peuvent être organisés hiérarchiquement selon la localisation géographique (cf. annuaire distribué).

5.2.3 Les SRICs

Les systèmes de recherche d'information coopérants (SRICs) sont des entités interrogeables soit par un utilisateur final, soit par un SRIC. Ils se scindent en deux catégories : les généralistes, les spécialistes.

Un SRIC possède les caractéristiques suivantes : la déclaration à un annuaire, la collecte d'informations sur le WWW, l'indexation des informations collectées, une interface d'accès à l'information ainsi qu'un protocole de communication.

La collecte d'informations

La collecte d'information peut s'effectuer de trois manières :

- Par parcours des liens hypertextes, à l'aveugle comme la majorité des outils de recherche. Cette méthode a l'avantage, en supposant le WWW connexe, de découvrir des nouveaux liens et des metadata.txt non référencées ou inconnues. Cependant, l'inconvénient est le même que celui des robots, à savoir l'obligation de télécharger la totalité des pages, et donc pas d'économie de bande passante.
- En s'adressant à un annuaire, et dans ce cas tous les SRICs déclarés seront alors connus.
- En interrogeant un SRIC, il s'agit d'une requête typique sur les metadata.txt du site hébergeant ce SRIC. Il est alors possible de récupérer les

descripteurs du site, ainsi que les descripteurs et accointances du SRIC en question.

Bien entendu, tout SRIC peut combiner les méthodes de collecte.

Déclaration à un annuaire

Tout SRIC, spécialiste ou généraliste qui se crée peut, et cela est préférable, s'enregistrer auprès d'un annuaire afin d'être référencé et donc accessible auprès du plus grand nombre.

Indexation

Cette partie est particulière aussi bien à un spécialiste qu'à un généraliste.

Les généralistes

Les généralistes ont comme point de départ une liste d'URL qui correspond à l'ensemble des sites de départ à parcourir. Cette liste est constituée manuellement ou fournie par un robot automatiquement (par exemple les sites les plus importants en volume d'informations qui auraient été répertoriés par des outils connus).

Un généraliste comporte deux modules. Un module visible par l'extérieur, c'est-à-dire dont les informations peuvent être lues à travers un certain protocole, et un module interne, qui contient entre autres la connaissance qu'il a des autres SRICs ainsi que sa base de données interne. Un généraliste est accessible via le protocole HTTP et comprend le protocole SGP.

Dans leur partie interne, les généralistes possèdent :

- un fichier de configuration décrivant leur fonction, les schéma de méta-informations, ontologie, le langage et le protocole utilisés pour transmettre et demander de l'information, les adresses des spécialistes associés aux méta-informations.
- une base de données contenant les méta-informations indexées.

Les spécialistes

Les outils spécialistes comportent entre autres :

- un fichier de configuration décrivant leur domaine de spécialisation, le schéma des méta-informations, le langage et le protocole utilisés pour

transmettre et demander de l'information, les accointances avec d'autres robots spécialistes ou généralistes.

- une base de données contenant les méta-informations relatives à leur domaine

Le robot généraliste parcours les sites de la liste ainsi que les références à d'autres sites afin de récupérer et d'indexer les méta-informations. Pour chaque site parcouru, la procédure est la suivante :

- Demander le fichier de configuration, conf.txt s'il existe, du site. Ce fichier est à la manière de robot.txt accessible sous la racine et comprend les informations suivantes :
  - type_robot : outil spécialiste ou généraliste
  - localhost : nom de la machine hébergeant les informations
  - schéma : collection d'attributs et ontologie du modèle
  - description : description du spécialiste
  - sujet : sujet traité par le spécialiste
  - portée : arbre des connaissances (par exemple arbre de la CDD) ou réseau sémantique décrivant les documents stockés par le spécialiste.
  - volume : volume de la base de données des méta-informations
  - accointances : arbre de connaissances ou réseau sémantique associés à des descriptions externes au spécialiste local.

Lorsque le généraliste se connecte au site hébergeant le spécialiste, il télécharge le fichier de configuration du spécialiste, lit le fichier, conserve les informations décrivant le spécialiste, met à jour sa base de données sur le spécialiste, et éventuellement alimente sa base de données à l'aide des méta-informations stockées par le spécialiste.

Le généraliste continue à parcourir l'ensemble des URLs de sa liste, s'il rencontre un site comportant de la méta-information, il télécharge tous les documents comportant ces méta-informations et les met à jour dans sa base de données de manière incrémentale. Si le généraliste rencontre un site ne comportant pas de méta-informations, il est libre de l'indexer ou non selon son propre algorithme d'indexation. Le but est que de plus en plus de sites soient capables d'indexer les méta-informations, et que, au fur et à mesure, un généraliste soit capable de garder les méta-informations sur les sites qu'ils a indexés et celles tenues par un spécialiste, afin d'éviter de se connecter à des sites déjà indexés par ce spécialiste.

Lorsque le spécialiste se connecte à un généraliste, il lui pose une requête sur son domaine de spécialisation, filtre les réponses des différents spécialistes de son do-

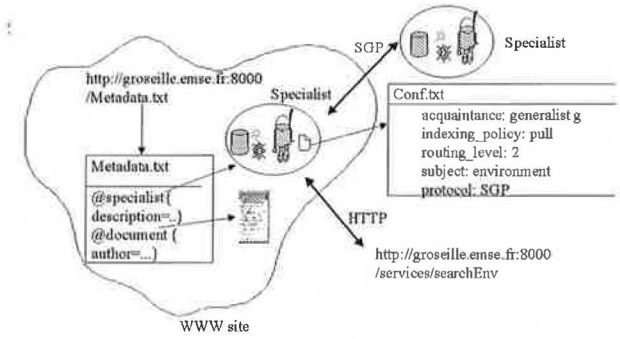


FIG. 5.7 – Elements d'un spécialiste.

maine et se connecte aux sites disposant de l'information pour leur prendre leur méta-information. Pour cela, il peut être guidé soit par une liste de spécialistes de son domaine, soit un spécialiste peut lui indiquer les autres sites ou spécialistes traitant de son domaine.

5.3 Architecture des outils spécialistes et généralistes

5.3.1 Rappel

Dans notre état de l'art, nous avons cité un certain nombre de protocoles dédiés à la recherche d'information (IR), tel que WAIS, et à ceux liés à une architecture d'annuaires distribués, tels que Whois++, CIP. Notre protocole s'intègre dans l'IR, dans la mesure où il s'agit de hiérarchiser des serveurs d'index qui contiennent, à la manière des centroides (concept défini dans Whois++), des informations regroupées dans un même domaine, ainsi qu'une connaissance des serveurs qui détiennent les informations propres. Ce protocole ne facilite pas uniquement le "routage de requête" (cf. CIP) mais plutôt permet de cibler les serveurs spécialistes intéressants pour répondre à une requête donnée, en faisant coopérer les différents outils d'une architecture distribuée.

5.4 Le protocole SGP

Le protocole S/G doit permettre d'implémenter l'envoi de messages entre les différents types de serveurs, chaque serveur est capable d'analyser une requête

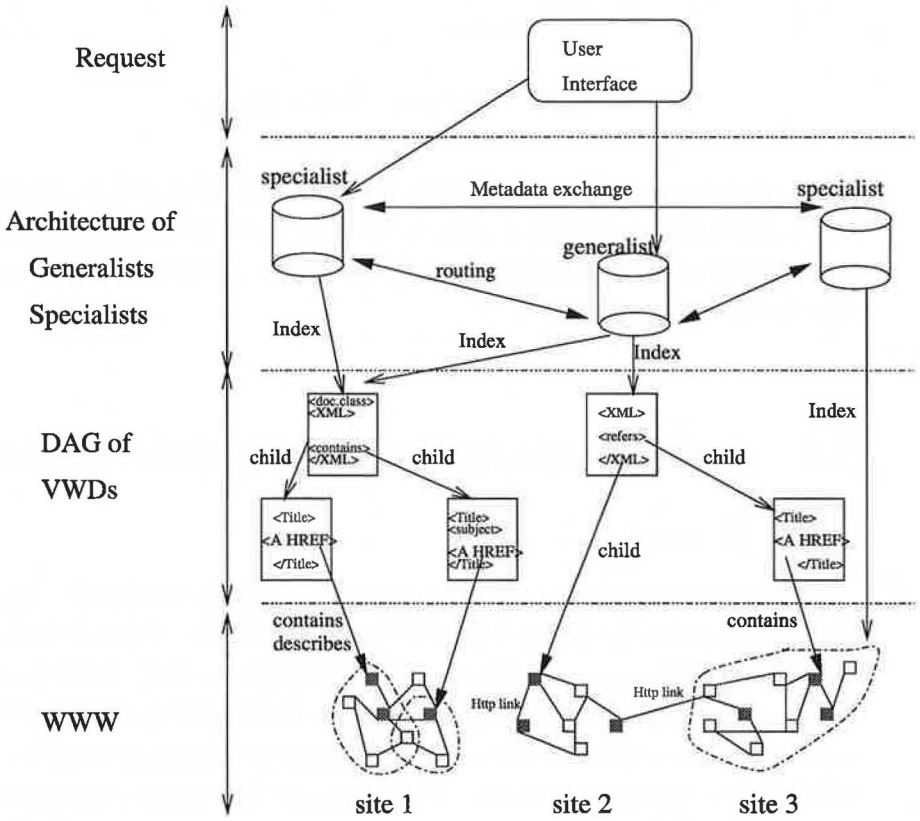


FIG. 5.8 – Architecture globale.

définie dans son protocole et de donner en réponse aux outils demandeurs les informations locales ou les serveurs susceptibles de répondre à cette requête.

Le protocole SGP est utilisé pour permettre aux serveurs généralistes et spécialistes de s'échanger leurs méta-informations, et de s'organiser en une hiérarchie ou DAG (graphe orienté acyclique) de serveurs d'index.
Nous considérons que différentes collections d'index peuvent décrire des bases de données hétérogènes, et que notre protocole est capable de transmettre tout type d'information entre des serveurs d'index, s'ils ont implémenté ce protocole. Nous spécifions l'architecture globale de SGP et les services rendus par les différentes entités. Puis nous définissons les messages et la syntaxe des index envoyés à travers le réseau. Nous expliquons en quoi le protocole que nous spécifions permet d'utiliser et de mettre en place des services facilitant la recherche d'information. Nous testons notre approche sur des serveurs de données locaux, pour différents thèmes.

Le but du protocole SGP est de fournir une structure des échanges d'informations plus souples que les structures définies par les protocoles étudiés précédemment, et ce afin que le système global tienne compte à la fois de l'évolution dynamique du WWW et des différentes vues que l'on peut avoir d'un même sous-espace d'informations. Au fur et à mesure de la construction des entités qui participent au modèle, les informations décrites par leur auteur ou par une autorité, l'expert d'un domaine par exemple vont s'auto-organiser autour des spécialistes qui vont structurer leur propre espace d'informations. Les entités vont garder ou créer de nouveux liens avec d'autres entités, afin de participer et coopérer à la recherche d'informations pertinentes pour une requête donnée. Le protocole SGP doit permettre d'implémenter l'envoi de messages entre les différents types d'entités, chaque entité est capable d'analyser une requête définie dans son protocole et de donner en réponse aux demandeurs les informations locales ou les serveurs susceptibles de répondre à cette requête. Selon la situation, une entité ayant une certaine autorité pourra router la requête aux entités dépendantes d'elle, et retransmettre les réponses à l'entité à l'origine de la demande.
Les objets échangés entre les différents outils de recherche représentent les méta-informations ou descripteurs des documents.

5.4.1 Méta-information

Nous avons opté pour une extension du format SOIF dans la description des documents ou pages en suivant les champs définis par meta data Dublin Core pour la sémantique. Ces méta-informations sont créées à différents niveaux :
Au niveau des sites WWW, il s'agit de fournir un minimum d'informations permettant d'aider les outils dans leur phase d'indexation de la sémantique des documents. Chaque document décrit est doté d'un identifiant unique :

```
@document{ document_id = d01.1
                         author(type = email) (MIME = RFCxxx) = brodhag@emse.fr
                         subject(scheme=CDD) = 333.3
                         keywords = environnement, pédagogie, entreprise
                         relation(type=child)=
                         http://groseille.emse.fr/ENVTEST/~brodhag/projelev
                         http://groseille.emse.fr/ENVTEST/~brodhag/d01.11 }
```

On notera que les documents sont organisés en DAG, les liens entre documents sont créés dans le processus de génération ou de mise à jour des documents. Soit il existe un outil de clusterisation automatique de documents avec création de méta-information pour les décrire ; soit l'auteur du site ajoute des tags META au sein des pages considérées comme points d'entrée aux documents. Dans ce cas, c'est l'outil spécialiste qui appliquera son propre algorithme de clusterization. Ainsi pour un même site, différentes organisations de l'information pourront être adoptées, selon les critères de plusieurs spécialistes d'un même domaine.
Si rien n'a été ajouté, les généralistes pourront indexer uniquement la page d'accueil du site parcouru.
Au niveau des spécialistes et des généralistes, chacun d'eux est tenu de se décrire de la même manière que les documents, c'est-à-dire en fournisssant un fichier metadata.txt accessible à l'adresse de leur serveur. Par exemple on peut avoir les renseignements suivants :

```
@specialist{ URL=http://groseille.emse.fr/ENVTEST/~brodhag/
             port=8000 subject(scheme=DDC) = 333.3
             description=specialiste sur l'environnement et entreprises }

@generalist{ URL=http://groseille.emse.fr/ENVTEST/~brodhag/
             language= En-US, Fr
```

```
port=9999 }
```

Cette méta-information est renseignée au niveau des sites qui alimentent les spécialistes, soit explicitement par un expert, soit au cours des échanges de données. Un spécialiste doit :

- Définir son domaine d'activité ou de spécialisation.
- Définir sa stratégie de recherche d'information et d'indexation. Définir un algorithme pour "clusteriser" les documents et répondre aux requêtes des utilisateurs ou de ses voisins.
- Définir son fichier de configuration qui détermine son profil (intérêts et compétences), la connaissance qu'il a de ses voisins, son protocole (formalisme des requêtes, interactions avec les autres robots), ainsi que la maintenance de son index de méta-information (par exemple, fréquence de mise à jour).

Un généraliste indexe les méta-informations trouvées sur le WWW. Il comporte :

- un fichier de configuration qui identifie son type, l'ensemble des robots spécialistes qu'il connaît, la fréquence de mise-à-jour des méta-informations.
- les moyens de stocker ces méta-informations et de les retrouver via un langage de requête.

Le spécialiste utilise des outils de l'IR (Information Retrieval) afin de répondre aux requêtes de son domaine de spécialisation. Un utilisateur pose une question à un robot généraliste qu'il connait, c'est au robot généraliste de s'adresser aux robots spécialistes concernés afin de répondre au mieux à la question posée.
L'utilisateur peut poser sa question directement à un robot spécialiste qui possède une interface d'interrogation. Nous avons défini les méta-information ainsi que le formalisme des messages envoyés par les différents types de robots dans le rapport [ ]. Nous détaillons ici les dialogues et interactions entre généralistes et spécialistes.
Prenons quelques exemples de requêtes utilisateur U à un généraliste, noté G et un spécialiste, noté S :

- U et S :
  S a des réponses, il renvoie ses réponses structurées, ou leur nombre suivant que la requête l'a spécifié.
  Si S connaît d'autres spécialistes de son domaine, il gère lui-même les fusions et les réponses.

S n'a pas de réponse, S détermine s'il est proche de la question, mais est trop général, dans ce cas s'il possède des liens avec des spécialistes plus spécifiques, il route la requête (cf. CIP). Il attend les réponses et les envoit à l'utilisateur final. Sa base de données pourra s'enrichir des réponses envoyées par les autres outils.
S n'a aucune idée de la requête, il se contente de renvoyer la réponse à un généraliste qu'il connaît ou donne une réponse négative s'il n'a pas d'adresse de généraliste.

- U et G :
  G donne en réponse à une requête les méta-informations des documents qu'il a indexés si elles correspondent ou bien des liens vers des spécialistes susceptibles de répondre mieux à la requête atttendue.

Initialisation d'un généraliste :
G parcours les sites WWW, soit à partir d'une liste d'URL référencées dans un fichier de configuration, soit en partant d'un site connu (Alta Vista, Lycos) ayant indexé le WWW. Il reconnaît les méta-informations et n'indexe que celles-ci.
Initialisation d'un spécialiste :
S parcourt les pages du WWW, à partir d'une liste de sites de son fichier de configuration, parcourt les métadata.txt et collecte les méta-informations qui concernent son domaine (spécifié dans ce fichier de configuration) ainsi que l'intégralité des pages WWW relatives aux documents. G se connecte à un site

S, télécharge les résumés de méta-information de S, et ajoute S à sa base de connaissance. Eventuellement, G vérifie que S n'existe pas déjà dans sa base, et si c'est le cas vérifie que les informations qu'il a déjà stockées sont à jour. S se connecte à un site G, donne son identité, pose une requête spécifiant quel est le langage utilisé et quel est son domaine de spécialisation. G répond par envoi et filtrage de méta-informations, met à jour sa base de connaisssance des robots spécialistes, peut éventuellement propager la requête à d'autres robots s'il ne peut y répondre directement et stocke éventuellement la requête envoyée par S. Les interactions sont formalisées :

```
open connection
 send request (@REQUEST)
```

```
  receive answer (list(@SUMMARY))
        close connection
```

Exemple de dialogue :

```
S envoie une requete a G:
 HEADER  : identite de S
type de requete
destinataire
 [schema de meta-informations]
[volume-max de donnees]
[date de derniere mise  jour]
 BODY    : contenu de la requete
 [niveau de priorite]
 [niveau de propagation]
[demande de mise a jour de G]

 G rpond a S:
 HEADER  : identite de G
 format de message
 [schema de meta-informations]
[volume de donnees]
BODY    : contenu des meta-informations
 [compte-rendu de priorite]
 [compte-rendu de propagation]
 [compte-rendu de mise a jour de G]
```

5.5 Implémentation

5.5.1 Prototype actuel

L'architecture actuellement en place permet de donner une idée des interactions entre un spécialiste et un généraliste et tient compte des contraintes dues à l'environnement du WWW. Typiquement les événements extérieurs qui modifient l'état de ce système sont :

- Prise en compte d'une requête d'un utilisateur,
- Annulation d'une requête,

- Apparition de nouvelles informations sur le WWW,
- Disparition ou modification d'informations sur le WWW.

La durée de vie d'une requête est limitée à un certain seuil. Si la réponse est donnée en dessous de ce seuil, elle sera affichée, sinon l'utilisateur est informé de l'échec de sa demande.

5.5.2 Spécification fonctionnelle

Nous détaillons à présent les diverses fonctionnalités de l'outil.

Collecte des informations sur le WWW

Chaque SRIC a sa propre politique de collecte de l'information, ainsi que d'indexation. Cette politique est précisée dans le fichier metadata.txt. Un SRIC utilise l'outil Harvest pour appliquer la politique du PULL, afin de créer les documents à partir des sites collectés par un robot qui parcours les pages WWW et un indexeur automatique qui extrait les balises et le texte associé aux pages WWW. La politique PUSH permet de faire une demande explicite d'indexation de pages à un SRIC, en fournissant les éléments nécessaires pour interagir avec le SRIC. Selon le cas, les spécialistes possèdent leur propre taxonomie de concepts, et peuvent ainsi classifier les documents qu'ils indexent selon cette taxonomie, donnant alors une autre vue des documents. Les méta-informations sont stockées dans une base de données, par exemple ORACLE ou bien dans un système de fichier géré par le spécialiste.

Saisie de la requête

Un formulaire HTML accessible à partir d'un navigateur doit permettre de remplir de façon conviviale les champs de méta-information, par exemple en associant aux attributs title, author leurs qualificateurs et les valeurs correspondantes. L'utilisateur doit également pouvoir spécifier d'autres informations, comme la profondeur de routage. À tout moment, l'utilisateur doit avoir loisir d'interrompre la requête.

Interprétation de la requête

Le système doit être capable de comprendre les requêtes au format HTTP et les traduire en requêtes SOIF. Ce format est utilisé pour uniformiser les communications internes au système.

```
@Request{  author(type = email) (MIME = RFCxxx) = brodhag@emse.fr
           subject(scheme=CDD) = 333.3
           keywords = environnement, pédagogie, entreprise
           scope: 2-level
           volume-max:
           volume-min:
}
```

scope donne le niveau de profondeur maximum de routage de la requête, alors que volume-max et volume-min sont des filtres sur les volumes des réponses renvoyées ou données à transmettre par un SRIC.

Traitement de la requête

Cette fonction consite à rechercher l'ensemble des pages et documents WWW qui s'apparient la requête. Afin de remplir cette tâche, les SRICs vont coopérer selon un protocole bien défini. Les objets échangés entre les SRICs sont au format SOIF, les documents WWW, les pages et les requêtes ont donc un identifiant unique.

5.5.3 Services

Un service décrit la liste de ses services à la demande d'un autre service. Il comprend les requête au format standard (SOIF en l'occurence). Il consulte sa propre base de données pour répondre à la demande. Il transmet la requête à d'autres services pour affiner la réponse. Il reconnaît une requête bouclante. Il décrémente la durée de vie de la requête lors de la transmission de celle-ci à un autre service. Il collecte les informations par push ou pull et enfin transforme les requête HTTP en SOIF et affiche les résultats. Les objets créés sont les suivants, pour implémenter les diverses fonctionnalités des services :

- service : écoute le réseau et analyse les différentes réponses
- outil d'indexation : cette classe se dérive en outil d'indexation du spécialiste et outil d'indexation du généraliste.
- base de donnée : consultation, mise à jour et analyse des réponses pour fusion ou suppression des redondances de réponses.
- routage : pour le routage des requêtes vers des serveurs dont le serveur principal est responsable. Contrôle du bouclage en supprimant les URLs

des serveurs déjà visités. Contrôle du time-to-live. Si le délai est dépassé, il s'arrête. Dans l'autre cas il route la requête. Le time-to-live est calculé par rapport au niveau de profondeur de spécialisation du spécialiste ayant fait l'objet de la requête.

- réponse service vérifie que la réponse est bien identifié comme provenant d'une requête valide dans le registre des requêtes, afin de transférer la réponse finale à l'utilisateur.
- réponse formate la réponse et la renvoie au serveur ou utilisateur originaire de cette requête.
- annulation supprime l'identifiant d'une requête de son registre des requêtes.

5.5.4 Fonctionnalités détaillées

Initialement, un service est en écoute. Lorsqu'on lui demande une connexion, il l'accepte. Il identifie ou non le type de communication (réponse, requête) avant de traiter la connexion sinon il ferme la connexion. Les différents types de connexions sont : demande de description (incluant son domaine d'intérêt), réponse envoyée par un autre service, requête d'un utilisateur, annulation d'une requête par l'utilisateur, requête envoyée par un autre service, demande de mise à jour de sa base de donnée.

Si le serveur reçoit une demande de description, il envoie son fichier metadata.class au serveur qui lui a demandé.

Si le serveur reçoit une réponse d'un autre serveur, il contrôle la validité de la réponse, i.e. il vérifie que la réponse correspond bien à une requête qui n'a pas été annulée entre temps ou pour la quelle on n'a pas fournit une réponse. Puis il renvoie les réponses soit vers le navigateur soit vers le service qui l'a demandé.

Afin de conserver l'historique de chaque requête client, les requêtes sont stockées dans un registre client requête. En fonction du calcul du temps de réponse maximum demandé par l'utilisateur et du calcul prévisionnel du temps de réponses des sous-services (dans la hiérarchie), soit les réponses actuelles sont envoyées à l'utilisateur, soit la requête est routée vers d'autres services.

L'annulation de la requête entraine sa suppression dans le registre.

Lorsqu'un serveur reçoit une requête d'un autre serveur il la traite de la même manière qu'une requête utilisateur excepté le fait qu'il récupère son identifiant.

5.5.5 Gestion de file d'attente

Lorsque le modèle s'adresse à plusieurs services, un même spécialiste peut se voir poser plusieurs requêtes, dans ce cas là, selon sa politique, il gère une file d'attente ou peut transmettre simultanément deux requêtes. De même, lors du routage, il est susceptible d'adresser une même requête à plusieurs spécialistes concernés, il a alors son propre algorithme (calcul du spécialiste le plus pertinent ?) pour séquencer les questions. Le problème se posera alors au moment de l'unification des réponses données par deux spécialistes différents.

5.5.6 Diagramme de flux

Le diagramme d'état et les flux entre les différents objects définis dans l'architecture procède comme suit :

L'objet Serveur est initialement en écoute. A la demande d'une connexion, il accepte. puis il se décrit et analyse la demande en fonction de la nature de la connexion, finalement il revient à l'état d'écoute.

L'objet Base de donnée consulte la base de donnée locale au SRIC. Puis il analyse les réponses obtenues afin de les filtrer ou les fusionner en cas de routage.

L'objet Réponse pour serveur formate la réponse et l'envoie au serveur ayant recu la requête correspondante d'un client.

L'objet Routage contrôle le bouclage, c'est-à-dire supprime les URLs des serverveurs déjà visités. Puis il contrôle le "time to live". Si le délai est dépassé il arrête. Dans l'autre cas et si cela est possible il route la requête vers un autre serveur.

L'objet Client enregistre la requête de l'utilisateur. Il lui attribut un identifiant et il le stocke dans son registre. Ensuite il crée un objet requête.

L'objet Réponse serveur contrôle la validité de la requête, il vérifie que son identifiant figure bien dans le registre des requêtes. S'il le trouve il transfère la réponse vers l'interface. Dans l'autre cas il s'arrête.

L'objet Annulation supprime l'identifiant d'une requête du registre des requêtes.

5.6 Conclusion

La mise en œuvre de cette architecture de SRICs a nécessité l'implémentation d'un protocole de communication entre SRICs et d'un mécanisme de synchro-

nisation permettant de gérer les transferts d'information.

L'avantage de ce système est d'alléger le volume d'informations transitant entre les SRIC et stocké dans chaque SRIC. De plus il permet à chaque serveur d'enrichir et de mettre à jour sa base de connaissance au fur et à mesure des interrogations. Ici nous avons le même système d'interrogation de la base de données pour chaque serveur. Dans les améliorations futures, il s'agirait de permettre à chaque serveur d'interpréter une requête selon son SGBD et d'avoir son propre mécanisme d'interrogation de sa base de données.

Chapitre 6

Conclusion

6.1 Synthèse et apports

Dans le chapitre bibliographique, nous avons répertorié deux grandes catégories d'outils de recherche sur le World Wide Web : Les annuaires thématiques, dont un exemple bien connu est YAHOO, qui permettent d'identifier des ressources Web se rapportant à un thème ou à un sujet précis sans connaissance préalable de ces ressources. Les moteurs de recherche en texte intégral, par exemple Google, Alta Vista, HotBot qui sont issus de l'application des techniques des SRI qui ne classent plus ces ressources mais indexent directement le contenu même de celles-ci. Or, ces types de moteurs ont deux défauts très importants : les annuaires thématiques en n'indexant pas le contenu des ressources Web ne sont pas capables de gérer des requêtes évoluées portant sur le contenu des ressources ; quant aux moteurs de recherche en texte intégral, ils ne disposent d'aucune information sémantique sur le contenu de ces ressources et ne peuvent donc prendre en compte le contexte associé à ces ressources lors des requêtes.

Pour résoudre les insuffisances des moteurs classiques et prendre en compte les problèmes intrinsèques au World Wide Web, à savoir l'hétérogénéité du corpus ainsi que la croissance exponentielle de son volume, nous avons défini dans cette thèse une nouvelle architecture de systèmes de recherche d'informations qui coopèrent en s'échangeant des informations modélisées grâce aux documents Web.

Notre solution a consisté à développer les deux points ci-dessous validés par un prototype.

6.2 Un modèle de documents Web

La motivation de ce modèle réside dans l'étude préalable de ce qui fait la particularité de notre système : le corpus matérialisé par le Web. Notre modèle de documents Web est une abstraction des pages HTML physiques, au sens où un document Web est défini par un ensemble de pages Web, ayant une sémantique propre, décrite grâce à un ensemble de méta-informations. Un schéma permet de déterminer comment sont constituées les méta-informations, c'est-à-dire quels sont les champs permis, leur complexité et leur signification. Un document Web appartient à un espace de documents Web organisés en DAG. Cet espace nous a servi à définir la notion de contexte attaché à un document Web (qui lui-même est un nœud du DAG), à savoir l'ensemble des nœuds antécédents de ce nœud. Ce modèle apporte plusieurs niveaux d'explicitation des données du Web :

- Au niveau sémantique

  Les méta-informations permettent de décrire en détail aussi bien des informations textuelles que des informations non textuelles, telles que les images, les fichiers informatiques en utilisant un vocabulaire contrôlé standard. Elle peuvent décrire aussi bien des collections génériques, à l'aide du schéma Metadata Dublin Core, que des données spécifiques (par exemple FGDC - Federal Geographic Data Comittee - a été élaboré en 1994 pour décrire des données sur les systèmes d'information de l'environnement) car le modèle permet l'extension de schéma.

  Plus précisément, les méta-informations décrivent dans quel domaine ou environnement se situe l'information (description interne) et fournit d'autres informations comme la version d'une page, à quel public l'information s'adresse, la certification d'une page par un expert, l'auteur des pages (information externe). Certaines méta-informations sont utilisées pour préciser la description de pages Web à l'aide de mots-clés. Ces méta-informations sont écrites dans les pages Web par leur auteur. Notre approche est différente, car elle a pour but d'introduire une description sur une page ou sur des ensembles cohérents et organisés de pages, plus adaptée aux organisations pré-existantes de sites Web. Nous avons utilisé les méta-information dans un cadre bien précis, à savoir les attacher non pas à chaque page du WEB (dont la structure hypertexte ne permet pas d'en extraire des informations facilement descriptibles) mais plutôt à des ensembles cohérents et organisés de pages.

  Alors que la classification de sites par Yahoo est figée et dépend d'une

classification de concepts unique, dans notre modèle nous avons plusieurs systèmes de classification afin de permettre la représentation de plusieurs vues d'un même ensemble d'informations. Dans ce sens, nous proposons le moyen d'expliciter la sémantique non seulement de n'importe quel type de données du Web, mais également à des niveaux de détail variable.

- Au niveau structurel

  Dans notre modèle de document Web, nous avons défini les liens structurels entre documents Web. Sachant que la majorité des outils de recherche classiques sur le Web ne tient aucun compte des liens hypertextes entre les pages Web pour en extraire un contexte, notre approche est radicalement opposée. Nous suggérons au contraire que la vue du Web n'est pas "plate", c'est-à-dire qu'il existe des liens structurels parmi les liens hypertextes, et qu'il nous faut les différencier des autres types de liens, et les rendre explicites. Nous mettons en évidence les liens structurels entre paquets de pages étiquetés que nous avons nommés documents Web.

- Au niveau contextuel

  Outre la notion de typage de liens que nous introduisons ici, nous définissons par le biais de la structure une hiérarchie de contextes, qui va permettre non seulement de visualiser un document Web ou une page Web dans son contexte mais encore de retrouver des documents en utilisant la propagation des attributs selon les liens de structure, telle qu'elle a été décrite dans la thèse de Fourel [Fou97], dans le cas des documents multimédia structurés.

  Par l'introdution des documents Web nous introduisons différents niveaux de granularité qui permettent une plus grande flexibilité des réponses et une autre manière d'appréhender les réponses, selon un point de vue qui peut être celui de l'auteur ou celui d'un expert dans le domaine.

  Nous donnons la possiblité de rendre l'organisation d'un site visible, à des niveaux de précision variables, et ce par l'auteur du site ou bien par un expert. De plus, non seulement les sites sont rendus visibles mais également par le même procédé, les outils de recherche spécialisés peuvent bénéficier de cette description et être mieux repérés.

  Enfin nous avons défini une grammaire de lanage permettant d'interroger les documents Web via un langage de requête de type SQL qui prend en compte le contexte et la structure, ce qui, à notre connaissance, n'est le cas d'aucun autre système de recherche d'informations actuel sur le Web.

- au niveau implémentation

Nous avons implémenté le modèle en utilisant XML qui est un langage très bien adapté à l'échange des méta-informations dans notre architecture. Par rapport aux outils existants, nous apportons une méthode pour décrire la sémantique de l'information sur le Web, en faisant abstraction de la contrainte physique des pages Web, à laquelle tous les outils de recherche sont soumis.

6.3 Une architecture de systèmes de recherche d'informations coopérants

En premier, l'hétérogénéité des ressources du Web, rend difficilement applicables les techniques classiques de la RI conçues pour des corpus homogènes. En second, l'explosion du nombre de pages pose des problèmes de "scalabilité" ou "mise à l'échelle" des bases de données des SRI. De plus la volatilité due aux fait que ces ressources sont générées de manière autonome, voire anarchique, ou plus simplement par leur nature (par exemple les forums de discussion) pose des problèmes de fiabilité des index et de mise à jour pour les SRI.

A ces deux problèmes, notre solution consiste à :

- Introduire de nouveaux outil pour gérer l'indexation autonome et distribuée de site.

  Répondre à l' évolutivité du Web en partageant l'administration des données indexées.

  Faciliter la mise à jour des données et résoudre le problème de scalabilité.

  Résoudre le problème de la sélection des données.

  Gérer l'hétérogénéité des ressources en proposant deux interfaces, l'une unifiéee pour répondre à des besoins généraux et sans connaissance de la structure existante donc indépendante de la structure, l'autre spécialisée pour des requêtes dépendant des outils de recherche et d'indexation.

  S'adapter à l'existant et aux outils présents sur le Web en tirant profit d'une meilleure description de l'information et des index.

  Nous répondons à un besoin qui n'est pas satisfait à l'heure actuelle, à savoir le partage des index par les différents indexeurs et la coopération par l'échange des index pour trouver une réponse adéquate.

  Répondre à un besoin de partage de l'information par une communauté, dans ce cas il est intéressant d'indexer les informations sur des descripteurs communs, et de parler le même langage (par exemple en utilisant

le vocabulaire d'un domaine, un thésaurus commun, une liste d'autorité, etc.)
Faciliter l'indexation et l'échange des données en fournisssant des listes d'attributs-valeurs lisibles par la machine, exportables, simples à maintenir.

6.4 Un prototype d'un système de recherche d'informations

Nous avons validé notre approche à travers le prototype d'un système de recherche d'informations qui prend en compte le modèle de documents Web que nous avons conçu. Dans le cadre de la création d'un spécialiste en environnement, des documents Web ont été créés par un documentaliste à partir du site francophone http ://www.Agora21.org/ dédié au développement durable. Ces documents Web explicitent la sémantique et la structure d'un site contenant plus de 2000 pages Web, hébergé sur le site de l'école des mines de Saint-Etienne. Les méta-informations sont décrites en XML, et la sémantique est explicitée à l'aide d'un thésaurus européen environnemental GEMET1.5 et de la classification CDD. Pour faciliter la création des documents Web, nous avons créé un éditeur muni d'une interface Web comprenant des champs de saisie. Le thésaurus GEMET ainsi que la CDD ont été mis en-ligne afin de permettre une interrogation des concepts au moment de la création des méta-informations. Le système de recherche d'informations comprend les modules suivants :

- Le module de collecte composé d'un robot qui alimente constamment la base d'index des pages à partir d'un certain nombre d'URLs données au départ, et locales à un site. Une interface de contrôle permet de spécifier la fréquence de collecte et le nombre d'ajouts dans la base du corpus. La base de données mySql, choisie pour stocker et récupérer les informations du corpus s'interface avec le langage Java. Des informations comme la date de création, le titre et l'URL de la page sont gardées dans la base de données. D'autre part, les adresses des documents XML sont également stockées dans cette base après analyse des champs à l'aide d'un parseur XML. Cette phase permet de constituer le corpus de notre système de recherche.
- Le gestionnaire d'index met en place les modules suivants :
  l'indexation est gérée par un fichier inversé des pages Web filtrées grâce

aux attributs de sélection stockées dans la base mySQL au moment de la constitution du corpus. Dans notre système, nous avons en réalité interrogé directement les pages sur leur contenu en utilisant la commande grep. En ce qui concerne les documents Web, un module de parseur XML analyse les documents pour en extraire l'arbre des attributs et pour les stocker dans une structure persistante. Nous utilisons un arbre pour stocker le DAG de documents, et nous propageons les valeurs des attributs afin de constituer la partie statique du système.

- Le gestionnaire de requête comprend :
  Le langage d'interrogation : nous traitons pour le moment des requêtes booléennes, avec une possibilité d'interroger sur le texte de la page et sur les attributs des documents Web. La fonction de correspondance : nous utilisons le modèle booléen, afin de déterminer quels sont les documents et les pages qui répondent à la question.
- Nous avons créé plusieurs spécialistes munis de documents Web fictifs pour simuler l'architecture des spécialistes et des généralistes, à un niveau de profondeur de trois maximum. Nous avons ainsi implémenté le routage des requêtes à partir d'une interrogation d'un généraliste vers deux spécialistes. Lorsque plusieurs requêtes arrivent, nous gérons une file d'attente.

6.5 Problèmes ouverts et limites

Nous terminons cete synthèse par les problèmes et les limites de notre système.

- La création des méta-informations est subordonnée aux décisons politiques et économique qui régissent le Web. Il est impossible d'imposer un schéma et un format de méta-informations universels, nous devons faire face aux problèmes de traduction de schéma qui sont aussi ceux rencontrés par les ontologistes.

- Il n'y a pas d'évaluation du modèle, pas de test avec TREC ou avec une autre méthode d'évaluation.

- L'architecture ne peut pas être testée à grande échelle, mais suppose que les spécialistes et les généralistes parlent le même protocole. Donc c'est plus dans des structures fermées de type intranet que nous pourrons utiliser ce système.

- Nous n'avons pas testé les performances de notre architecture par rapport aux autres architectures centralisées et distribuées.

- L'implémentation du modèle a utilisé le modèle booléen puis une fonction de classement des réponses mais nous pouvons envisager d'autres modèles plus compexes.

6.6 Perspectives

- Nous avons envisagé une automatisation de la reconnaissance de la structure des documents Web, qui consitute une voie de recherche en cours dans notre équipe.

D'un autre côté des recherches prometteuses trouver une grammaire de site pour définir la structure immédiatement par comparaison, mais encore au stade de prototype sont entreprises pour extraire automatiquement la structure et la sémantique de ces ressources, toujours de manière à utiliser la puissance des techniques de la RI classique. Il s'agit d'intégrer des techniques plus évoluées pour les adapter dans notre architecture, par exemple la logique floue ou une interface en langage naturel.

- une autre voie de recherche est l'intégration vers une plateforme multi-agents pour gérer les conflits et les dialogues entre spécialistes et utilisateurs

- utiliser l'inférence des ontologies pour faciliter l'apprentissages des spécialistes et des généralistes.

- Nous pouvons envisager d'étendre le langage de requête aux documents multi-média et affiner la structuration vers des éléments internes aux pages.

Bibliographie

[AD99] Fernando Aguiar and Bich-Liên Doan. Impact of the structure of documents in information retrieval. Technical Report 99.9, Centre SIMMO, Dpt. RIM, Ecole Nationale Supérieure des Mines de Saint-Etienne, 158, cours Fauriel 42023 Saint-Etienne Cedex 2, France, decembre 1999.

[ADB99a] Fernando Aguiar, Bich-Liên Doan, and Michel Beigbeder. Dealing with structured documents in information retrieval systems. In *WebNet99*, Honolulu, USA, October 1999.

[ADB99b] Fernando Aguiar, Bich-Liên Doan, and Michel Beigbeder. Deducing the context hierarchy of web sites. Technical Report 99.8, Centre SIMMO, Dpt. RIM, Ecole Nationale Supérieure des Mines de Saint-Etienne, 158, cours Fauriel 42023 Saint-Etienne Cedex 2, France, novembre 1999.

[ADB99c] Fernando Aguiar, Bich-Liên Doan, and Michel Beigbeder. Deducing the context hierarchy of web sites for information retrieval systemsl. In *WebNet99*, Honolulu, USA, October 1999.

[AL98] Paul Albitz and Cricket Liu. *DNS and BIND, third edition.* O'Reilly, 1998.

[And00] Olivier Andrieu, septembre 2000. http ://www.abondance.fr.

[Bar00] Christophe Bardy. Tout savoir des moteurs de recherche. *INFO PC*, (173) :72–83, septembre 2000. http ://www.infopc.fr.

[BBB+97] R. J. Bayardo, W. Bohrer, R. Brice, A. Cichocki, J. Fowler, A. Halal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. The InfoSleuth Project. In Joan M. Peckman, editor, *Proceedings, ACM SIGMOD International Conference on Ma-*

nagement of Data : SIGMOD 1997 : May 13–15, 1997, Tucson, Arizona, USA, volume 26(2) of *SIGMOD Record (ACM Special Interest Group on Management of Data)*, pages 543–545, New York, NY 10036, USA, 1997. ACM Press.

[BBB+98] R.J Bayardo, W. Boher, R. Brice, A. Cichocki, and J. Fowler. *InfoSleuth : Agent-based semantic integration of information in open and dynamic environments.* Morgan Kaufmann, 1998.

[BDH+95] C. Mic Bowman, Peter B Danzig, Darren R. Hardy, Udi Manber, and Michael F. Schwartz. The harvest information discovery and access system. *Computer Networks and ISDN Systems,28 :119-125*, 1995.

[Bet93a] Annie Bethery. *Abrégé de la classification de Dewey.* Edition du cercle de la librairie, 1993.

[Bet93b] Annie Bethery. *Abrégé de la classification décimale de DEWEY.* Paris, 1993.

[BL96] Peter Bruza and Mounia Lalmas. Logic-based information retrieval : Is it really worth it ? In *Proceedings of EUFIT 96, Fourth European Congress on Intelligent Techniques and Soft Computing*, Aachen, Germany, September 1996.

[BR99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley, 1999.

[CBC+00] R.G.G. Cattell, Douglas K. Barry, Rick Catell, Mark Berler, Jeff Eastman, David Jordan, Craig Russell, Olaf Schadow, Torsten Stanienda, and Fernando Velez. *The Object Data Standard : ODMG 3.0.* Morgan Kaufmann, 2000.

[CDU98] Édition abrégée de la classification décimale universelle, 1998. 6e édition.

[CF99] Jérome Charron and Christian Fluhr. Befor (beyond forms). In HERMES, editor, *H2PTM'99*, pages 27–41, 1999.

[CGRU97] Chandra Chekuri, Michael Goldwasser, Prabhakar Raghavan, and Eli Upfal. Web search using automated classification. Sixth International World Wide Web Conference (WWW6), 1997. Poster presentation.

[CK97] Y. Chiaramella and A. Kheirbek. *An integrated model for hypermedia and information retrieval*, chapter 7, pages 139–178. Kluwer

Academic Publishers, maristella agosti and alan smeaton edition, 1997.

[DAAP98] R. Dolin, D. Agrawal, A. El Abbadi, and J. Pearlman. Using automated classification for summarizing and selectingheterogeneous information sources. Technical Report january98-dolin, D-Lib Magazine, january 1998.

[DAB99] Bich-Liên Doan, Fernando Aguiar, and Michel Beigbeder. Expliciting structural and semantic links to improve information retrieval. In *WebNet99*, Honolulu, USA, Octobre 1999.

[DB99] Bich-Liên Doan and Michel Beigbeder. Virtual www documents : a concept to explicit the structure of www sites. In *IRSG99, 21st Annual Colloquium on IR Research*, Glasgow, Scotland, Avril 1999.

[DBGJ98] Bich-Liên Doan, Michel Beigbeder, Jean-Jacques Girardot, and Philippe Jaillon. Utiliser les méta-informations pour décrire les documents www : application aux échanges d'informations. In *NTICF'98*, Rouen, France, Novembre 1998.

[DBGJ99] Bich-Liên Doan, Michel Beigbeder, Jean-Jacques Girardot, and Philippe Jaillon. A model of a self-configuring organization to improve information retrieval on the www. In *KRIMS II (Second International Workshop on Knowledge Representation for Interactive Multimedia Systems)*, trento, Italie, Juin 1999.

[ddwi00] digital design works inc., 2000. http ://www.ddwinc.com/docs/pages.html.

[Dew76] Melvil Dewey. Catalogs and cataloging : a decimal classification and subject index. Technical report, U.S. Bureau of Education. Public Libraries in the United States of America : special report, part I, Washington, DC, U.S, 1876.

[Fa97] Myron Flickner and all. *Intelligent multimedia information retrieval*, chapter 1, pages 7–22. American Association for Artificial Intelligent, 1997. "Query by image and video content : the QBIC system".

[FAD+99] Dieter Fensel, Jurgen Angele, Stefan Decker, Michael Erdmann, Hans-Peter Schnurr, Steffen Staab, Rudi Studer, and Andreas Witt. On2broker : semantic-based access to information sources at the WWW. In *webnet99*, 1999.

[Fou97] Franck Fourel. Impact de la structure du document sur la recherche d'information. *INFORSID, Ingénierie des systèmes d'information*, 5 :339–366, 1997.

[Fou98] Franck Fourel. *Modélisation, indexation et recherche de documents structurés*. PhD thesis, Université Joseph Fourier, Grenoble, France, février 1998.

[Gua94] Nicola Guarino. The ontological level. In B. Smith R. Casati and G. White, editors, *Philosophy and the cognitive sciences*, 1994.

[GW95] J. Gargano and K. Weiss. RFC 1834 : Whois and network information lookup service, whois++, August 1995. Status : INFORMATIONAL.

[HA99] Bernardo A. Huberman and Lada A. Adamic. Evolutionary dynamics of the world wide web. *Nature*, 401(131), 1999.

[HHL99] Jeff Heflin, James Hendler, and Sean Luke. SHOE : A knowledge representation language for internet applications. Technical Report CS-TR-4078, University of Maryland, College Park, October 1999.

[HOY+99] Fumio Hattori, Takeshi Ohguro, Makoto Yokoo, Shigeo Matsubara, and Sen Yoshida. Socialware : multiagent systems for supporting network communities. *Communications of the ACM*, 42(3) :55–61, March 1999.

[HP93] Marti A Hearst and C. Plaunt. Subtopic structuring for full-length document access. In *Proceedings of the 16th annual international ACM SIGIR Conference on research and development in information retrieval, Pittsburgh.*, pages 55–68, 1993.

[HPPL98] Bernardo A. Huberman, Peter L. T. Pirolli, James E. Pitkow, and Rajan M. Lukose. Strong regularities in World Wide Web surfing. *Science*, 280(5360) :95–97, 1998.

[Inc00] Google Inc., june 2000. http ://www.google.com/pressrel/pressrelease26.html.

[JB97] R. Jr and W. Bohrer. Infosleuth : Agent-based semantic integration of information in open and dynamic environments. 1997.

[Khe95] A. Kheirbek. *Modèle d'intégration d'un système de recherche d'informations et d'un système hypermédia basé sur le formalisme des graphes conceptuels–Apllication au projet RIME.* PhD thesis, Université Joseph Fourier, Grenoble, France., 1995. PhD thesis.

[KKA] Hermann Kaindl, Stefan Kramer, and Luis Miguel Afonso. Combining structure search and content search for the World-Wide Web. In Kaj Grønbæk, Elli Mylonas, and III Frank M. Shipman, editors, *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia (HYPER-98)*, pages 217–224.

[KLW95] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4) :741–843, July 1995.

[Kor95] Robert R. Korfhage. VIBE : Visual information browsing environment. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Systems Demonstrations : Abstracts, page 363, 1995.

[Kor97] Robert R. Korfhage. *Information Storage and Retrieval.* John Wiley & Sons, 1997.

[LC95] T. Berners Lee and D. Connolly. Hypertext markup language - 2.0, 1995.

[Les64] Michael Lesk. The smart automatic text processing and document information retrieval system. Technical Report Report ISR-8, sec II, Harvard Computation laboratory, Cambridge, Massachusetts, 1964.

[Les97] Michael Lesk. *Practical Digital Libraries : books, Bytes, and Bucks.* Morgan Kaufmann, 1997.

[LLD96] Carl Lagoze, Clifford A. Lynch, and Jr. Daniel, Ron. The warwick framework : A container architecture for aggregating sets of metadata. Technical Report TR96-1593, Cornell University, Computer Science, June 21, 1996.

[Mae94] Patties Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7) :31–47, July 1994.

[Mau96] Hermann Maurer. *HyperWave, the next generation Web solution.* Addison-Wesley, 1996.

[Mil98] Eric Miller. An introduction to the resource description framework. *D-Lib Magazine*, May 1998. http ://www.dlib.org/dlib/may98/miller/05miller.html.

[OV91] M. Tamer Ozsu and Patrick Valduriez. *Principles of distributed database systems.* Prentice Hall, Inc., 1991.

[PIC] Platform for internet content selection. *The World Wide Web Consortium (W3C).*

[RFC81] Internet protocol. by Information Sciences Institute University of Southern California for Defense Advanced Research Projects Agency Information Processing Techniques Office, 1981. RFC 791.

[RFC87] Domain name. concepts and facilities. Network Working Group, 1987. RFC 1034.

[RFC93] Fyi in what is the internet. User Services Working Group de l'Internet Engineering Task Force (IETF), 1993. RFC 1462.

[RFC96] Hypertext transfer protocol – http/1.0. Network Working Group, 1996. RFC 1945.

[Rij79] C.J. Van Rijsbergen. *Information Retrieval.* 1979.

[RW96] Mark A. Sheldon Chanathip Namprempre Peter Szilagyi Andrej Duda David K. Gifford Ron Weiss, Bienvenido Velez. Hypursuit : A hierarchical network search engine that exploits content-link hypertext clustering. *Proceedings of the Seventh ACM Conference on Hypertext, Washington, DC*, 1996.

[SA94] G. Salton and J. Allan. Text retrieval using the vector processing model. In *Proceedings of the third annual symposium on document analysis and information retrieval*, pages 9–22, Las Vegas, Nevada, 1994.

[Sal64] Gerard Salton. A flexible automatic system for the organization, storage, and retrieval language data (smart). Technical Report Report ISR-5, sec I, I. Harvard Computation laboratory, Cambridge, Massachusetts, 1964.

[SF97] John R. Smith" and Shih-Fu Fang. *Intelligent multimedia information retrieval*, chapter 2, pages 23–41. American Association for Artificial Intelligent, 1997. Querying by color regions using the visualseekcontent-based visual query system.

[SK98] Amit P. Sheth and Wolfgang Klas. *Multimedia Data Management : Using Metadata to Integrate and Apply Digital Media.* McGraw-Hill, 1998.

[SM98] Ralph Steyer and Olivier Manesse. *HTML 4.0.* Guide de l'utilisateur ; 2053. Micro Application, Paris, France, 1998.

[SW95] Eric Miller Ron Daniel Stuart Weibel, Jean Godby. Oclc(online computer library center)/ncsa(national center for supercomputing applications) metada workshop report. *The essential elements of network object description*, 1995.

[W3C] W3C. Resource Description Framework. `http://www.w3.org/RDF/`.

[WGMD95] Stuart Weibel, Jean Godby, Eric Miller, and Ron Daniel. OCLC/NCSA metadata workshop report, March 1995.

[XML00] Extensible markup language (xml) 1.0 (second edition), october 2000. http ://www.w3.org/TR/2000/REC-xml-20001006.

Annexe A

Les modèles associés aux fonctions de correspondance

A.1 Le modèle booléen

Ce modèle est utilisé par la majorité des outils de recherche et est représenté par :
Soit un document D représenté par un ensemble de termes (mots ou expressions composées). Une requête booléenne est une expression logique composée de termes connectés par les opérateurs booléens ET, OU, ou SAUF. Les documents retrouvés ou non par le système sont déterminés par la présence ou l'absence des termes spécifiés dans la requête. Cette technique d'appariement strict entre les requêtes et les documents est encore très utilisée dans les systèmes commerciaux de recherche d'information en partie parce que l'implémentation de cette méthode à l'aide des fichiers inversés est simple et robuste. Cependant l'inconvénient majeur de ce système réside dans le caractère dichotomique des réponses, c'est-à-dire qu'un document répond ou non à une requête, les documents donnés en réponse ne sont pas classés par ordre de pertinence, par exemple par rapport à une mesure de similarité effectuée sur la fréquence d'occurence de termes. Nous étudions dans ce qui suit deux modèles bien connus qui suivent cette approche.

A.2 Le modèle vectoriel

Les utilisateurs fournissent une expression ou une phrase (ensemble de termes non connectés par des opérateurs booléens), le système leur renvoie des réponses classées par ordre de pertinence, en tenant compte de la fréquence d'occurence des termes. Ce modèle a l'avantage de donner en réponse des documents qui ne contiennent par forcément tous les termes de la requête, et donc est tolérant aux erreurs, et peut gérer des requêtes complexes difficilement traduites avec le modèle booléen.

Le système SMART est le premier SRI basé sur le modèle vectoriel, développé à l'université d'Harvard [Sal64], [Les64]. Ce modèle repose sur des mesures de similarités entre une requête et un document ou entre deux documents.

Soit un corpus de documents utilisant un ensemble n de termes uniques T_i. Les documents du corpus ainsi que les requêtes sont représentés par des vecteurs dans l'espace à n dimensions, affectés respectivement à des coefficients de poids t_i et q_i. Soit un document $D = t_1T_1 + t_2T_2 + ... + t_nT_n$, et une requête $Q = q_1T_1 + q_2T_2 + ... + q_nT_n$, La mesure la plus largement utilisée est la mesure du cosinus (Van Rijsbergen 1979) déterminé par le cosinus entre D et Q :

$$Cos(D,Q) = \frac{\sum_{i=1}^{n} t_i q_i}{\sqrt{\sum_{i=1}^{n} t_i^2 \sum_{i=1}^{n} q_i^2}} \quad \text{(A.1)}$$

Plus un document est pertinent à une requête, plus le cosinus entre les deux vecteurs est proche de 1. Les documents répondant à la requête sont classés par odre de pertinence. Dans le système SMART, le calcul de la proximité entre un document et une question permet de reformuler la requête. Les termes des documents jugés pertinents par l'utilisateur prennent un plus fort poids t_i ou sont rajoutés à la requête alors que les termes jugés non pertinents sont diminués de leur poids ou supprimés de la requête.

Néanmoins, dans ce modèle les termes des documents sont traités indépendamment les uns des autres alors que dans le langage naturel ce n'est pas le cas. Le modèle suivant tente d'apporter une amélioration dans ce problème. Il se base sur l'assertion suivante :
Les documents jugés pertinents à une requête doivent avoir un poids plus fort que les documents non pertinents.

A.3 Le modèle probabiliste

L'idée de ce modèle est qu'un document réponde de manière pertinente à une requête si la probabilité P(Rel/D) que le document D soit pertinent est strictement supérieur à la probabilité P(Nrel/D) que le document soit non pertinent à la requête. La recherche probabiliste [Kor97] a donné de bons résultats en précision et rappel, cependant, la masse de calcul et d'assertions faites n'ont pas justifié son utilisation par rapport aux performances comparées à l'emploi des modèles vectoriel et booléen. Cette voie a donc été peu suivie par les développeurs de SRI. D'autres modèles, comme le modèle connexioniste ou les réseaux neuronaux ont également fait l'objet d'étude mais la contrainte de performance est à prendre en compte aussi ces modèles sont-ils utilisés pour des besoins très spécifiques.

A.4 Le modèle de correspondance floue

Le problème soulevé par le modèle probabiliste réside dans l'estimation des probabilités. La correspondance floue est similaire à la correspondance probabi-

liste à la différence que l'estimation probabiliste est remplacée par l'estimation qu'un document est pertinent à une requête. L'idée de requête floue suggère que pour une requête spécifique, on croit qu'un document est pertinent. La notion de document floue par contre n'existe pas, il n'y a pas de degré d'appartenance de termes à un document. Cependant un jugement flou peut être fait sur la pertinence d'un document à une requête.

A.5 Le modèle basé sur la logique

La RI basée sur la logique est apparue il y a une dizaine d'années. Son objectif était d'améliorer l'efficience(précision et rappel) en apportant une "sémantique" au processus de recherche. Les modèles basés sur la logique ont fait leur preuve à la fois sur le plan théorique et sur le plan pratique [BL96]. A la différence de la majorité des systèmes fondés sur le calcul de la fréquence d'occurrence des termes dans un document, les modèles fondés sur la logique utilisent la sémantique pour améliorer la recherche d'information. Un modèle théorique sous-tend la sémantique du modèle logique. Soit M un modèle et ϕ une formule, alors $M \models \phi$ affirme que ϕ est vraie dans M, ou M satisfait ϕ. La signification de ϕ dans le contexte de M est réduite à une valeur vraie. Dans l'IR, si q est une formule exprimant un besoin d'information, et le document d une structure dans laquelle q est interprétée, on suppose que si $d \models q$ alors d a un rapport avec la requête q et donc doit être retournée en réponse à l'utilisateur. Cependant si on transpose ce modèle à un SRI, la différence réside dans la représentation du document en son index $\chi(d) \models q$ ne correspond pas à l'exacte vérité, mais ne nous intéressons-nous pas à l'à-propos d'un document sur la requête ? Deux types d'approches répondent à ce modèle : les approches de situation théorique et les approches des mondes possibles.

Annexe B

le langage XML

B.1 Description du langage XML

Les éléments textuels et multimedia afin d'être combinés, échangés et publiés doivent s'inscrire dans une organisation, XML fournit le support d'une telle organisation. XML est un format adapté pour stocker des données structurées et semi-structurées.

```
<transaction>
<time date= " 19980608 "/>
<from id= " 56743 "> BL Doan </from>
</transaction>
```

Un document XML a une structure logique et une structure physique. La structure logique permet de décomposer un document en unités et sous-unités appelées élements. La structure physique permet de stocker les composants appelés entités d'un document dans différents fichiers physiques. XML est un métalangage, il décrit d'autres langages. Il n'y a pas d'éléments prédéfinis, l'utilisateur peut en créer à souhait. XML utilise une DTD (Document Type Definition) pour définir les éléments autorisés dans un type de document. Il permet donc un contrôle de la structure logique d'un document. Chaque document possède des règles définies dans la DTD pour déterminer sa validité. Exemple de DTD :

```
< !ELEMENT warning (para*)>
< !ELEMENT para    (#PCDATA|keyword)*>
< !ELEMENT keyword (#PCDATA)*>
```

Il est possible de créer un éditeur pour créer et éditer des données XML, comprendre la DTD. XML tend à encourager le contenu de l'objet plutôt que l'affichage comme nom d'élément. Mais comment présenter le paragraphe? Une feuille de style est nécessaire. Plusieurs feuilles de styles sont définies pour un ensemble d'éléments. XML (version 1.0 en fevrier 1998) est construit sur la base du langage SGML(ratifié par l'ISO en 1986), hérite de quelques caractéristiques de HTML et inclue de nouvelles caractéristiques. HTML ne fournit pas les méchanismes permettant à l'auteur de rajouter leurs propres tags. Le futur de XML est presque garanti, d'abord parce qu'il y a un réel besoin d'un " Markup Language simplifié ", ensuite parce qu'il n'a pas de concurrents et est relié à SGML et HTML .

Applications avec XML XML est appliqué dans différents domaines comme la technologie push, l'EDI (échange de données électroniques) et les métainformations (MCF (Netscape), XML-Data (Microsoft), RDF (Netscape)). Un

serveur peut interpréter des données XML, répondre aux opérations requises par une applet java. XML peut être utilisé comme format d'échange pour des données stockées dans des bases de données. XML peut être utilisé pour la publication de documents semi-structurés (catalogues, manuels utilisateurs etc...) :

```
<book CommData= " 2345667 ">
<chapter>
<title>Example de document HTML </title>
<note>
<para> premier paragraphe </para>
</note>
</chapter>
```

Un titre de chapitre est son réel contenu lorsqu'il apparaît en haut du chapitre mais est une méta-information lorsqu'il est référencé comme contenu d'une page. Il est facile d'extraire du XML pour le stocker sur différents supports, grâce à son format structuré et controllé. XML inclue d'autres options standards, comme XLL pour les liens hypertextes et XSL pour le format de sortie. CSS (Cascading Style Sheet) peut s'utiliser comme langage pour décrire un format de document HTML et XML.

Pourquoi nous choisissons XML ? C'est un langage simple qui s'auto-décrit. Chaque auteur peut rajouter ses propres tags et décrire sa DTD pour décrire la structure logique d'un document. Il peut s'insérer dans HTML. Il existe de nombreux parseurs et XML permet un contrôle important sur un type de document créé. XML est maintenant compris par Internet Explorer et Gecko en version beta de Netscape.

B.2 Balisage de document

Un document entier est enfermé dans un élément de document. La structure hiérarchique d'un livre par exemple est représentée par un arbre. Les relations père, fils, frère existent entre les différents éléments. Les structures hiérarchiques peuvent ête récursives. Un élément contient directement des instances de lui-même. La granularité désigne le degré auquel le contenu d'un document est découpé en éléments fils. Un attribut permet de stocker les méta-informations. Optionnellement un document est déclaré comme suit :

```
< !DOCTYPE MyBook SYSTEM " MyBook.DTD ">
ou < !DOCTYPE MyBook [
```

```
.................
]>
suivi d'une instruction (Processing Instruction) pour identifier la version uti
< ?XML version= " 1.0 " ?>
```

Une entité peut être partagée et référencée par plusieurs documents.L'auteur d'un document peut s'aider d'entités pour construire ce document. Il faut déclarer une entité :

```
< !DOCTYPE MyBook [
< !ENTITY MyEntity " content of MyEntity "....>
< !ENTITY MyEntity " content ignored"....>
]>
```

Déclaration d'un paramètre d'entité :

```
<!ENTITY  % AnEntity " (para|list) ">
<!ENTITY  AnEntity " This is an entity ">

Pour les entités externes, la ressource est identifiée par :
<!ENTITY MyEnt SYSTEM "http ://www.XMLserve.com /ENTS/MYENT.XML ">
```

La structure logique (DTD) fournit les règles de définition de la structure d'un document. Certaines déclarations sont stockées dans le document, d'autres à l'extérieur.

B.3 Hypertext links (XXL)

Les liens de références croisées référencent une ressource, la cible à un élément lié, la source. Les attributs source et cible peuvent partager une localisation

```
See <link target="http ://Myserver.org.com/XML/doc#X1213">More information </li
Doc :
<chapter ident="X1213">
<title>chapter 3</title>
In this chapter , this is detailed
</chapter>
```

Les liens peuvent être bi ou multi-directionnels, par exemple : Napoléon est relié a sa biographie, ses conquêtes, la psychologie.

B.4 Typage des liens en XML

Pour retrouver l'information dans un livre ou dans une de ses structures équivalentes on peut utiliser :

- les références aux numéros de pages,
- les numéros de section ou de chapitre,
- un index de termes,
- une table des matière.

Les liens hypertextes "Pour plus d'informations" sont utilisés afin d'arriver à la section spécifiée. XML offre une convention pour identifier une ressource par sa localisation contextuelle. Il est possible d'identifier une catégorie à laquelle appartient un lien en attachant un rôle à un lien. Par exemple

```
<link href=" #123" title="location">
```

, pourra être exploité par des navigateurs spécialisés. Les "extended links" peuvent être stockés à l 'extérieur des documents. Différents types de liens existent : Les liens simples sont uni-directionnels. Lorsque la DTD n'est pas utilisée, il faut spécifier la valeur de l'attribut xml-link.

```
<simple href="http"
xml-link="simple">
See section
</simple>
```

Annexe C

Implémentation des SRICs

C.1 Éditeur de documents web (WeDoc)

Un éditeur de documents web a été développé dans le cadre de cette thèse. A travers une interface graphique écrite en Perl sous la forme de CGI script, il permet de créer des documents Web en précisant les différents attributs et qualificateurs d'un document Web. Il autorise également la création de liens structurels entre ces documents, en créant des sous-documents à partir d'un document. Afin d'indexer les documents Web, un thésaurus multlingue européen dédié au développement durable est accessible en ligne. Les copies décran suivantes sont respectivement l'éditeur de création des documents Web, l'éditeur de la description de ces documents Web grâce à des méta-informations et le thésaurus GEMET1.5.

Pour le concepteur du site ou l'expert documentaliste, il permet de :

- construire les documents web à l'aide de filtre qui permet d'extraire automatiquement les pages de l'arborescence du site.
- associer les méta-informations aux documents ainsi créés.
- naviguer entre les documents à travers de liens hypertextes représentant les relations structurelles entre documents web.

Cet outil génère un ensemble de fichiers ASCII, qui peuvent ensuite être convertis en XML ou SOIF. Le fichier XML est ensuite déposé à la racine du site documenté, et pourra être accédé par des Collecteurs.

Le site AGORA21 a été ainsi entièrement décrit en utilisant cet outil. voici un extrait du fichier "métadata-xml" généré par l'outil :

```
<?xml version="1.0"?>
<!DOCTYPE METADATA SYSTEM "http://groseille.emse.fr:2806/metadata.dtd">
<METADATA>

<DOCUMENT>
  <URL>#d1</URL>
  <CREATOR>agora21</CREATOR>
  <SUBJECT SCHEME="DDC-21">363.7</SUBJECT>
  <SUBJECT SCHEME="GEMET-1.5">8247</SUBJECT>
  <SUBJECT>d{\'e}veloppement soutenable, francophone, francophonie</SUBJECT>
  <DESCRIPTION>site francophone du d{\'e}veloppement durable.\npr{\'e}sence des textes des Conven
  <RELATION TYPE="CHILD">#d9,#d13,#d16,#d20,#d29,#d32,#d33,#d37,#d38,#d42,#d56,#d77,#d78,#d83,#d8
  <RELATION TYPE="MEMBERS">
      http://framboise.emse.fr/WWW/www.agora21.org/dd/.index.html
      http://framboise.emse.fr/WWW/www.agora21.org/dd/brundtland.html
      http://framboise.emse.fr/WWW/www.agora21.org/dd/bareg.html
      http://framboise.emse.fr/WWW/www.agora21.org/dd/dates.html
```

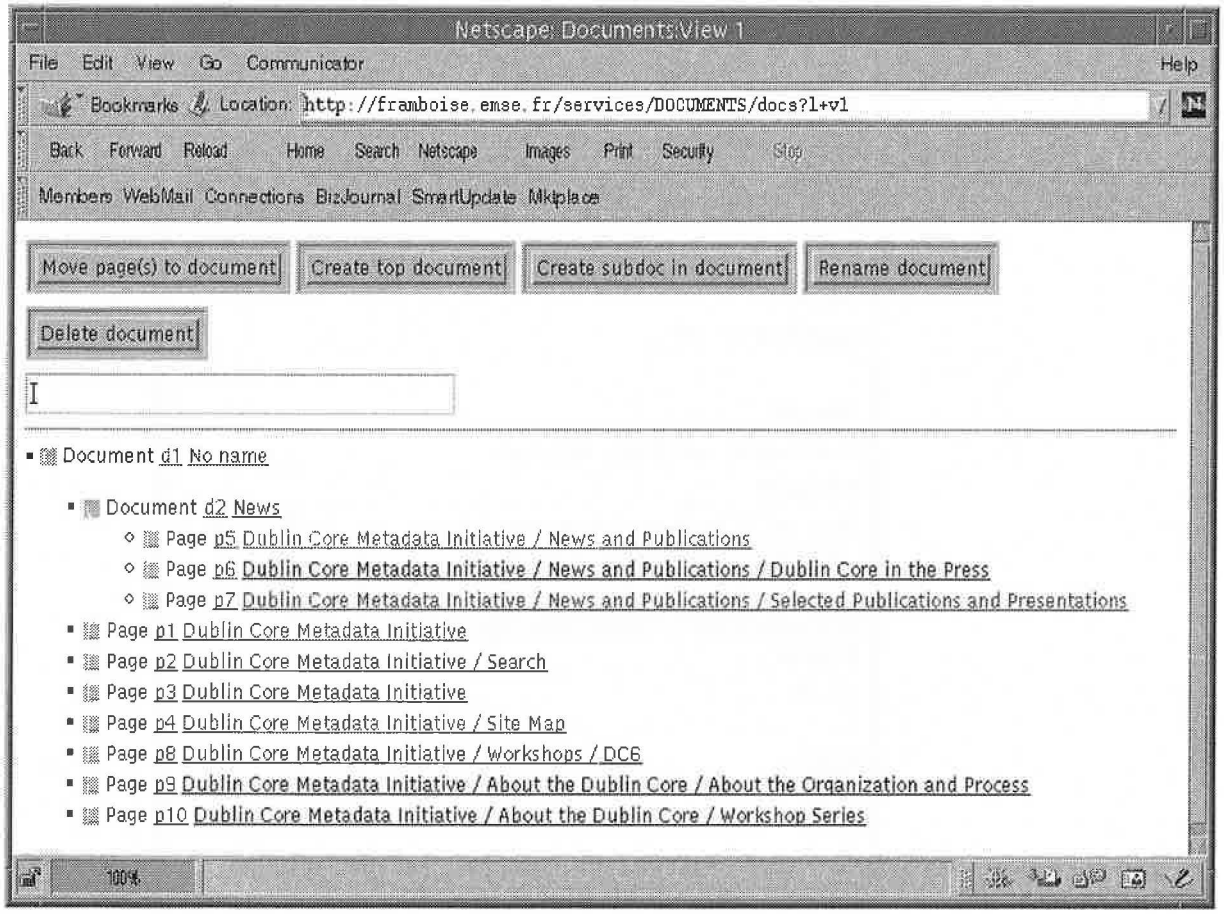


FIG. C.1 – Editeur de documents Web

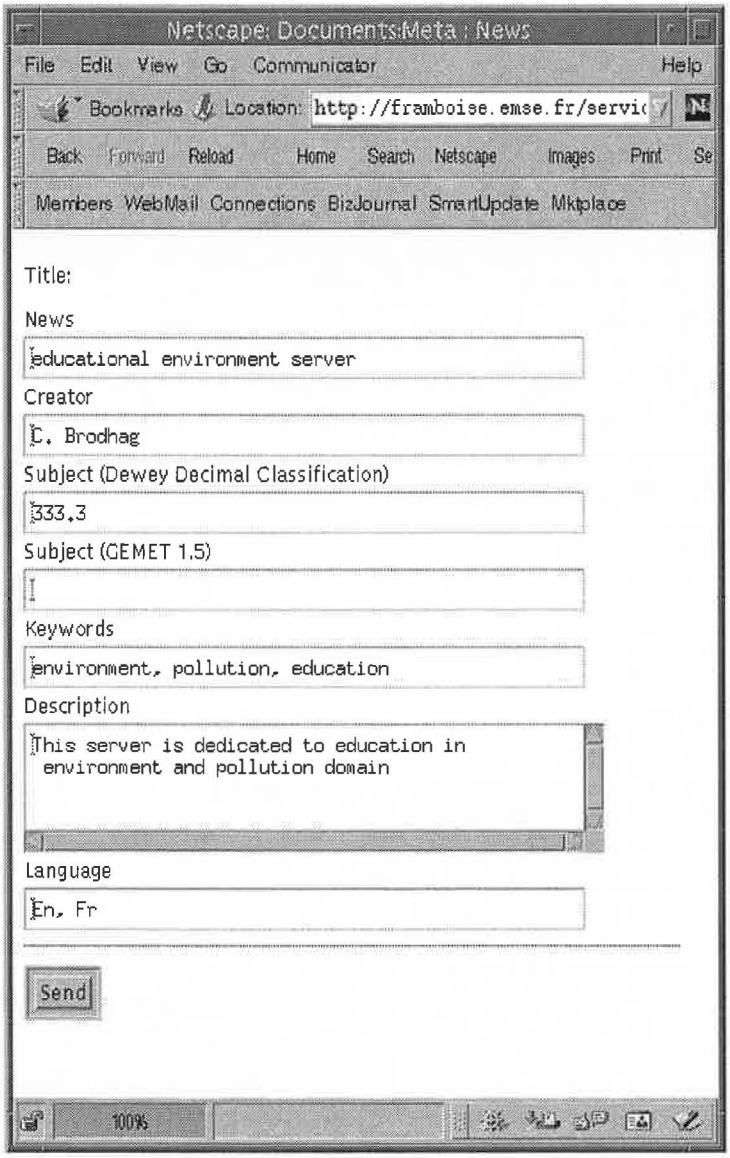


FIG. C.2 – Editeur de méta-informations pour décrire des documents Web

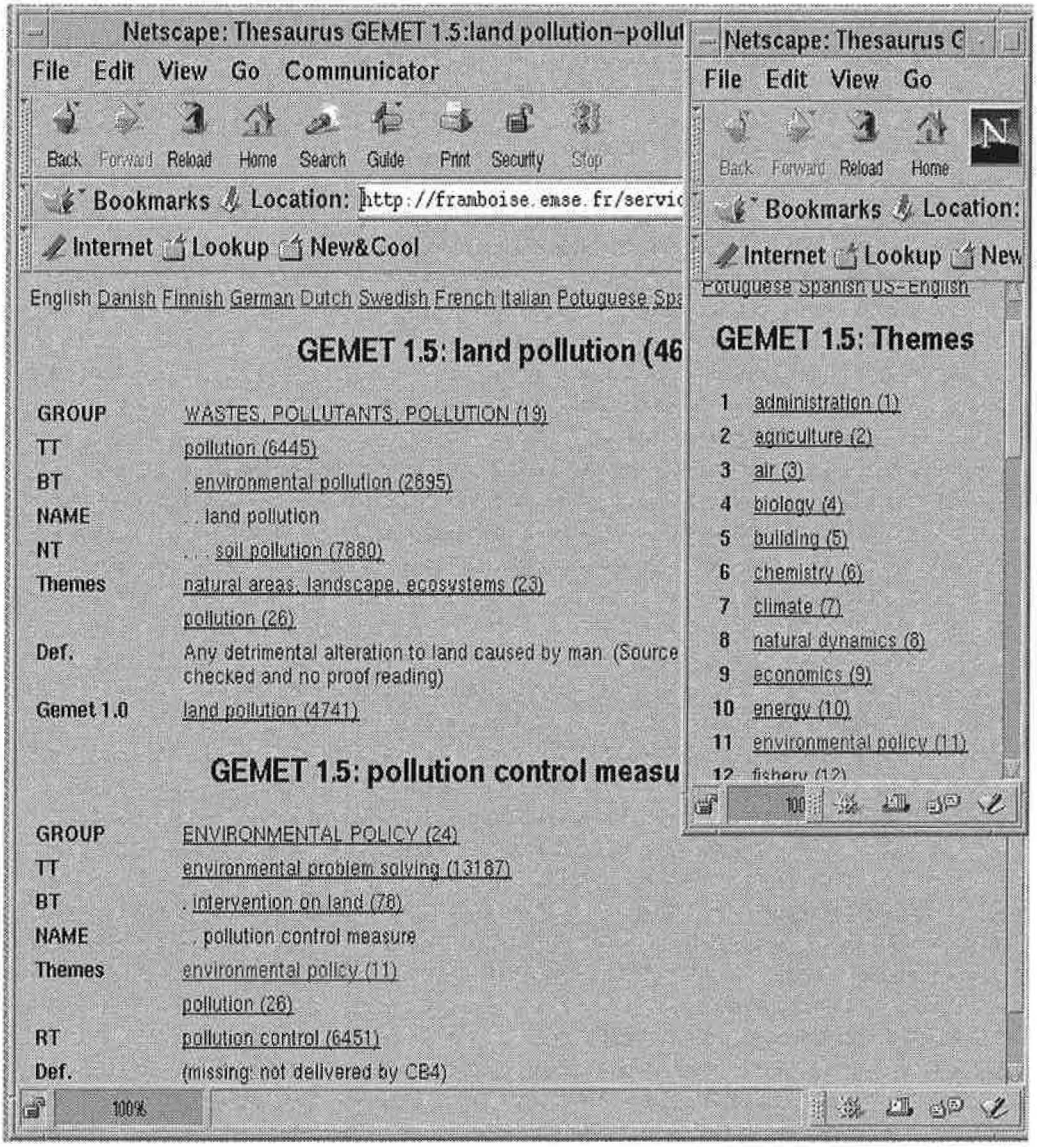


FIG. C.3 – Thésaurus GEMET 1.5

```
      http://framboise.emse.fr/WWW/www.agora21.org/cites21/pres2.html
      http://framboise.emse.fr/WWW/www.agora21.org/nouveau2.html
  </RELATION>
</DOCUMENT>

<DOCUMENT>
  <URL>#d9</URL>
  <CREATOR>agora21</CREATOR>
  <SUBJECT SCHEME="GEMET-1.5">13756</SUBJECT>
  <SUBJECT>ville durable, charte</SUBJECT>
  <RELATION TYPE="CHILD">#d10,#d11,#d12</RELATION>
</DOCUMENT>

<DOCUMENT>
  <URL>#d10</URL>
  <CREATOR>agora21</CREATOR>
  <RELATION TYPE="MEMBERS">
      http://framboise.emse.fr/WWW/www.agora21.org/aalborg/index.html
      http://framboise.emse.fr/WWW/www.agora21.org/aalborg/aalborg02.html
      http://framboise.emse.fr/WWW/www.agora21.org/aalborg/aalborg00.html
      http://framboise.emse.fr/WWW/www.agora21.org/aalborg/aalborg01.html
  </RELATION>
</DOCUMENT>

<DOCUMENT>
  <URL>#d11</URL>
  <RELATION TYPE="MEMBERS">
      http://framboise.emse.fr/WWW/www.agora21.org/aalborg/frame02.html
      http://framboise.emse.fr/WWW/www.agora21.org/aalborg/frame01.html
      http://framboise.emse.fr/WWW/www.agora21.org/aalborg/frame00.html
  </RELATION>
</DOCUMENT>

<DOCUMENT>
  <URL>#d20</URL>
  <CREATOR>agora21</CREATOR>
  <SUBJECT SCHEME="GEMET-1.5">8776</SUBJECT>
  <SUBJECT>ONU, assembl{\'e}e g{\'e}n{\'e}rale, programme de travail, CDD</SUBJECT>
  <DESCRIPTION>texte de l'AG de l'ONU de 1997 suivi du programme de travail de la CDD pour les pr
  <LANG>fran{\c c}ais</LANG>
  <RELATION TYPE="CHILD">#d26,#d99,#d101</RELATION>
</DOCUMENT>

<DOCUMENT>
  <URL>#d32</URL>
  <SUBJECT SCHEME="DDC-21">529.3</SUBJECT>
```

```
  <SUBJECT>manifestation, colloque, conf{\'e}rence</SUBJECT>
  <DESCRIPTION>pr{\'e}sentation et liste des diff{\'e}rents calendriers( nationaux et internationaux).pr{\'e}senta
  <LANG>fran{\c c}ais</LANG>
  <RELATION TYPE="MEMBERS">
     http://framboise.emse.fr/WWW/www.agora21.org/calendrier/.index.html
      http://framboise.emse.fr/WWW/www.agora21.org/calendrier/calendrier.html
      http://framboise.emse.fr/WWW/www.agora21.org/calendrier/dates.html
      http://framboise.emse.fr/WWW/www.agora21.org/calendrier/frame1.html
      http://framboise.emse.fr/WWW/www.agora21.org/calendrier/frame3.html
  </RELATION>
</DOCUMENT>
```

C.2 Les robots

Dans notre implémentation nous distinguons deux catégories de robots :
- les robots collecteurs : qui rapatrient localement les documents web.
- les robots analyseurs : qui indexent ces documents web.

C.2.1 Les robots collecteurs

Écrit en java, le robot collecteur recherche les URL des documents web au format XML, les analyse et récupère ces document web ainsi que les pages HTML associées. Ces informations sont alors stockées dans une base de données SQL via de JDBC. Cette base de données contient une description des documents web (les attributs et qualificateurs des documents, les URLs des pages html associés) ainsi qu'un ensemble d'information de collecte : date de création des documents, dernière mise à jour, etc permet de gérer les délais de rafraîchissement de ces informations.

C.2.2 Les robots analyseurs

A partir de la base de données créée par les robots collecteur, les robots analyseurs extraient les documents pertinent par rapport à leurs domaines d'indexation, récupèrent les pages HTML pour une analyse textuelle et créent la base d'indexe du spécialiste. Ils propagent également les attributs des documents de manière à implémenter efficacement les requêtes prenant en compte le contexte dans ces documents. Cet outil est également écrit en Java.

C.3 Une interface de requête

Une interface de requête existe également écrite en Java. cette interface accessible soit sous forme d'application ou d'applet java permet de naviguer dans la base documents du spécialiste, de visualiser les pages html associées à ces documents web. Elle permet surtout d'introduire des requêtes sur les documents en explicitant les valeurs souhaitées pour les attributs et qualificateurs des documents recherchés ainsi que le texte devant figuré dans les pages html associées à ces documents. L'interface interroge alors la base d'indexe et affiche les résultat dans une fenêtre et permet également de visualiser les pages ainsi retrouvées.

C.4 Des spécialistes et généralistes

L'interface de requête s'appuie sur un moteur de spécialiste/généraliste également écrit en Java. Ces spécialistes et généraliste peuvent dialoguer entre eux via un protocole communication utilisant le langage XML. Le rôle des spécialistes et généralistes ici consiste uniquement à matcher les requêtes avec la base de données constituée par les robots analyseurs et renvoyer les résultats à l'interface de requête ou les forwarders à d'autres spécialistes/généralistes.

C.5 Les bibliothèques

Les applications citées ci-dessous s'appuie sur des packages écrit en java dont voici les plus importants :

- `dogs.docwww.index` : ce package renferme les classes java implémentant la représentation des documents ainsi que le fonction de propagation des attributs.
- `dogs.gui` : ce package renferme les classes relatives à l'interface graphique de l'interface de requête.
- `dogs.applet` : ce package contient la version applet de l'interface de requête.
- `dogs.docwww.parser` : ce package contient les classes relatives au robots collecteurs et robots annalyseurs et notamment la gestion de la base de données SQL via le JDBC.
- `dogs.docwww.sric` : ce package implémente les protocoles de communication entre spécialistes et généraliste avec un niveau de routage.